OPTIMAL CONTROL OF FERMENTATION PROCESSES

By

Gabriel Eduardo Carrillo-Ureta

A Thesis submitted for the Degree of

Doctor of Philosophy

City University, London

Control Engineering Research Centre

School of Engineering

Northampton Square

London EC1V 0HB

United Kingdom of Great Britain

March, 2003

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I wish to express my sincere thanks to my supervisor Professor Peter D. Roberts for his help throughout these four years. A special thanks to Dr. Victor Becerra for his unconditional assistance. Without them this could not be possible.

Also thanks to all my good friends in the Electrical Engineering Department: Daniel, Lackson (R.I.P.), Ziad, Stavros, Ermioni and Moufid because they have been great friends during my PhD life. Not forgetting Joan and Linda for their good advises and support.

My sincere gratitude to all my family in Panama, in particular: my father Daniel, my mother Vielka and my sister Michelle for always supporting me in everything that I do; and happiness for the newest member of our "next generation", my niece Melanie. A very special mention to my girlfriend Amy for her support during the final study years, her encouragement and patience will always be deeply appreciated. Also thanks to Ing. Roberto Barraza and Dr. Emir Humo in the Universidad Tecnologica de Panama for their valuable advise.

But above all, thanks to God for leading my steps every day of my life and giving me the opportunity to be where I am now; without Him these studies could not be possible at all; thanks for everything.

DECLARATION

The author grants power of discretion to the University Librarian to allow the thesis to be copied in whole or in part without further reference to the author. This permission covers only single copies made for study purposes, subject to normal conditions of acknowledgement.

ABSTRACT

The general purpose of this thesis is to focus on a particular industrial process (from the beer industry) which serves as a guidance example for optimal control using different algorithms/methods. At the same time, the aim is to demonstrate the capabilities/features of MATLAB and SIMULINK as tools used in programming algorithms and simulation for optimal control of non linear systems. The thesis shows how to approach an optimisation problem with different techniques and to compare them on the same basis.

The main reasons for carrying out research on a beer fermentation process can be summarised as follows: this kind of industry represents an up-to-date example of industrial processes in general, the need to compare and evaluate optimisation methods (well established and "modern") on similar circumstances using the same process model and finally, give a good foundation for the control engineer to follow-up this work with different optimisation techniques and/or any other industrial process.

The fundamental features of the methods used involve the viability of known previously tested algorithms for optimal control of beer processes with high non-linearity and constraints; thus testing the flexibility of some of the known MATLAB Toolboxes for the optimal control of a particular simulated mathematical model.

An important aspect of the experimentation that has been carried out, is the creation of a simulated model of a selected beer process by means of including the mathematical equations, parameters and initial conditions into an s-function block. This SIMULINK model also incorporates the particular objective function that can be calculated directly after the simulation of the process for a particular input temperature profile. Together with the use of some available MATLAB functions for the formulation of particular optimal control techniques, this facilitates the creation of program routines that can be interfaced with the simulated process.

The final results using different optimisation methods such as: the gradient method in function space, DISOPE algorithm, Genetic Algorithms and Sequential Quadratic programming; show substantial improvement in the perfomance index obtained. The optimised temperature profiles found can be implemented for industrial application to provide a maximised ethanol production under particular restrictions, i.e. final by-products concentration, contamination risk and brisk changes in temperature.

LIST OF ABBREVIATIONS

Abbreviation	**Explanation**
ARX	Auto regressive eXternal input
CPU	Central Procesing Unit
DISOPE	Dynamic Integrated System Optimisation and Parameter Estimation
DMS	Dimethyl Sulphide
EOP	Expanded Optimal control Problem
EQP	Equality constrained Quadratic Programming
FPE	Final Prediction Error
GA	Genetic Algorithms
HGA	Hybrid Genetic Algorithm
IQP	Inequality constrained Quadratic Programming
KKT	Karush-Kuhn-Tucker
LP	Linear Programming
MB	Mega Bytes
MHz	Mega Hertz
MO	Multi-Objective
MOP	Model based Optimal control Problem
MMOP	Modified Model based Optimal control Problem
NLP	Nonlinear Optimisation Problem
NN	Neural Networks
OBS	Optimal Brain Surgeon
PC	Personal Computer
PEM	Prediction Error Method
QP	Quadratic Programming
RAM	Random Access Memory
ROP	Real Optimal control Problem
SO	Single-Objective
SQP	Sequential Quadratic Programming
TSP	Travelling Salesman Problem
VDK	Vicinal Diketone

CHAPTER I

INTRODUCTORY TERMS AND BASIC CONCEPTS

Beer fermentation including some basic concepts is going to be the main focus of this chapter. Definitions of some introductory terms will help to understand the parameters related to this work. Optimal control techniques are also a main point of discussion and will be introduced for an initial understanding on how optimisation techniques work.

1.1 AIMS AND OBJECTIVES OF THE THESIS

The main aims and objectives of the research can be summarised as follows:

- To improve the results obtained previously by Andres-Toro et al (1997a) concerning the optimisation of the beer process model, herewith obtain an even superior objective function value. Together with this, show how other optimisation techniques can provide a better understanding of the process behaviour.
- To present a useful introduction on the importance of selecting an appropriate mathematical model as the starting point for optimal control, this without considering a too complicated (that cannot be properly optimised) or a too simple one (that does not follow the real behaviour closely).
- To design general MATLAB scripts for different optimisation techniques that are simple, flexible and adaptable for the optimal control of any industrial process simulated from a mathematical model.
- To obtain an implementable optimised temperature profile that can be used in the beer making industry. This can be achieved by applying a smoothing technique when needed to make the profile functional.
- To make a comparison of some known (also considering new and/or modified approaches) optimisation methods and techniques under similar conditions and

give some advice on how to choose the most suitable algorithm for a particular process.

A list of the literature reviewed from previous work related to this thesis has been included as an initial reference to some background information:

- The main beer fermentation model considered for optimal control has been created from previous work by Andres-Toro et al (1998).
- The book "Optimal Control in Fermentation Processes" by Leigh (1986) has been used for consultation and review at the beginning of this research.
- Other mathematical models of fermentation processes have been considered useful in the first stages of the research as an insight on different approaches to model real industrial processes. A review on these papers has been included in the MPhil to PhD transfer report (Carrillo-Ureta, 1999).
- For every particular optimisation technique/method, papers on the different algorithms used and its application to particular cases have been reviewed: the principles behind the Gradient Method in Function Space have been considered from the book by Noton (1972); for the case of the DISOPE algorithm, work by Roberts (1993) and Becerra and Roberts (1995); the use of the Genetic Algorithms Toolbox by Chipperfield et al (1999) has been reviewed from work by Chishimba (1998); and finally, the Sequential Quadratic Programming algorithm under MATLAB has been examined from the Optimization Toolbox Manual (Coleman et al, 1999).

1.2 INTRODUCTION TO BEER FERMENTATION

In order to produce beer, water and barley are mixed together to generate a sweet infusion, the wort; this is then blended with hops and later fermented with yeast. This may look as a simple procedure but in practice it can be extremely complicated. Yeast is a single celled microorganism that transforms the sugar in the wort into alcohol and carbon dioxide (it reproduces by maturing). There are actually hundreds of varieties and strains of yeast and in general each brewery has its own variety, which principally determines the particular beer character.

The term "fermentation" is derived from the Latin verb *fervere*, that means to boil, thus describing the appearance of the action of yeast on extracts of fruit or malted grain (Stanbury et al, 1995). For some yeast varieties, the cells go up to the top at the end of the fermentation, giving the name of top fermentation (ales are brewed like this). On the other hand, when at the end of fermentation the yeast cells fall to the bottom, bottom fermentation is achieved (used for lager or pils). Earlier, there were just two types of beer yeast: ale yeast (the top-fermenting type: *Saccharomyces cerevisiae*) and lager yeast (the bottom-fermenting type: *Saccharomyces uvarum*, previously known as *Saccharomyces carlsbergensis*). Nowadays, as a result of recent reclassification of Saccharomyces species, both ale and lager yeast strains are considered to be members of *Saccharomyces cerevisiae*.

Fermentation has come to have different meanings to biochemists and to industrial microbiologists. Its biochemical meaning relates to the generation of energy by the catabolism of organic compounds, whereas, its meaning in industrial microbiology tends to be much broader. Brewing and the production of organic solvents may be described as fermentation in both senses of the word but the description of an aerobic process as fermentation is obviously using the term in the microbiological context.

Even though beers are brewed from related resources, beers throughout the world have their own individual styles. Their distinctiveness comes from the mineral content of the water used, the types of ingredients employed, and the difference in brewing methods. Nonetheless, it can be said that there are two established beer styles: ales and lagers. However, in addition to ales and lagers, there are other

classical beer styles such as wheat beers, porters, stouts, and lambics that are worth mentioning. Albeit most of the classical beer styles originated in Europe (Belgium, British, Czech Republic, French, German and Irish just to mention the most important), many of them are brewed effectively all over the world. With any beer technique there are no unbreakable policies and variations within flavour, ingredients, and methods of brewing for different styles are to be expected. Brewmasters each have their own explanation of what they consider suitable for the style.

According to Goldammer (2000), malt influences the flavour of beer more than any other ingredient: the malt types selected for brewing will determine the final colour, flavour, mouth feel, body, and aroma. As regards on the style of beer desired and the type of malt, it takes from 15 to 17 kg of malt to produce a hectolitre of beer. There is no general structure used for classifying malts since "maltsters" classify and advertise their products in their own way. Though regularly malts are classified as: base malts (Pilsner malt, Pale Ale malt and Mild Ale malt), specialty malts (light-coloured: Vienna malt and Munich malt; dark-coloured: Amber malt and Brown malt), caramelised/crystal malts (Dextrin malts), roasted malts (Chocolate malt and Black malt), unmalted barley (Roasted barley), and other malted grains (Wheat malt and Rye malt).

Wheat malt, understandably, is essential in making wheat beers. Wheat is also used in malt-based beers for the reason that its protein gives the beer a pleasant mouth sensation and improves beer head stability. As a negative aspect, wheat malt contains significantly more protein than barley malt, around 13 to 18% more, and consists primarily of glutens that can result in cloudy beer. European wheat malts have generally a smaller amount of enzymes than American malts, perhaps because of the malting methods or the varieties of wheat used.

The conventional way for beer fermentation is to add yeast to the worth and wait for some time, letting the yeast consume substrates and produce ethanol (without stirring). According to the industry, lager yeast strains are best used at temperatures ranging from 7 to 15°C. Herewith, lager yeasts develop slower than ale yeasts, and with less surface foam they tend to settle down to the bottom of the fermentor close

to the end of the fermentation (referred to as bottom yeasts). The ultimate flavour of the beer depends significantly on the strain of lager yeast and the temperatures at which it was fermented. Thus, fermentation can be accelerated with an increase of temperature but, however, some contamination risks (Lactobacillus, etc.) and undesirable by-products (diacetyl, ethyl acetate, etc.) could appear.

Defining the flavour and aroma of beer can be very difficult, this arises as a result from a large selection of parameters that come up from a number of different sources. Malt, as mentioned before, hops, and water have a great impact on the beer flavour, the decomposition of yeast which forms by-products during fermentation and maturation, is in addition essential (Goldammer, 2000). The most distinguished of these by-products are certainly ethanol and carbon dioxide; however additionally, a great amount of other flavour compounds are also formed:

a) Esters are among the most significant aroma components in beer, regarded of imparting a fruity quality to beer. As indicated by the production companies, esters are more desirable in ales than in lagers. Ester production can be increased by several factors including: high fermentation temperatures; restricting wort aeration; increasing the attenuation limit; and also increasing the wort concentration. Additionally, the type of yeast has an effect on ester concentration.

b) Diacetyl can be classified as a ketone and has also an important significance to beer flavour and aroma. Together with another ketone 2,3-pentanedione, these are described as the *vicinal diketone* (VDK) content of beer, which is the primary flavour in differentiating aged beer from green beer. Even so, diacetyl is critical because it is produced in larger quantities and also having a larger influence in the beer flavour than 2,3-pentanedione. The existence of diacetyl manifests with a buttery flavour typically, whereas 2,3-pentanedione has more of a honey taste. The flavour of diacetyl can firstly be confused with the taste of caramel malts, but afterwards it can be distinguished: diacetyl frequently tends to be unbalanced in most beers, revealing a rough flavour; the flavouring imparted by caramel malts conversely, is likely to be stable.

c) More than a few active aldehydes can affect the beer flavour. Aldehydes in beer can be produced during brewing at different periods by means of the oxidation of alcohols and various fatty substances. They are reduced to ethanol by the end of the primary fermentation. If oxygen is introduced back into the process, then the ethanol is oxidized back into acetaldehyde. Kunze (1996) affirms that the production of acetaldehyde can be increased by different factors: a fast fermentation; rise of the temperature throughout the fermentation; low wort ventilation; and infected worts (particularly by varieties of *Zymomonas anaerobia*). Just like with diacetyl, the incidence of active yeast in the maturation stage is a requirement for adequately little aldehyde levels in the final product. By means of a warmer maturation phase, aldehydes can be avoided with ease.

d) Organic and inorganic sulphur volatiles for instance hydrogen sulphide, dimethyl sulphide, sulphur dioxide, and thiols can have an effect on the beer flavour. Sulphur compounds can be tolerable or still desired if present in lesser amounts, but can increase to distasteful off-flavours (rotten egg flavour) in larger concentrations. Three important basic ingredients of sulphur in beer are: untreated resources (malt and/or hops), yeast metabolism, and spoilage organism.

e) Another wanted flavour compound in lager beer (not desirable in ales) and accountable for malty/sulphuric taste is the dimethyl sulphide (DMS). DMS can, in addition, improve the malt integrity of the beer. It is known in the industry that malting process itself have more repercussion on beer DMS quantities than the conditions surrounding the fermentation. It is also acknowledged that reduce fermentation temperatures and increasing wort gravity help DMS production.

f) Fusel alcohols are another kind of by-products that play a part directly to beer flavour (sometimes referred to as higher alcohols). Also important because of their involvement in ester formation. They enclose strong flavours, producing an "alcoholic" or "solvent-like" scent (known to leave a warming

effect on the palate). According to its production, the yeast strain seems to be very significant, some being able to produce up to three times as much fusel alcohols as others. Another factors that increase the production of fusel alcohols are: high fermentation temperatures, mutant yeasts, high wort gravities, intensive aeration of the pitching wort, and low amino acid concentration in wort.

g) Organic acids can be derived from malt and are present at low levels in wort. nonetheless, their concentration increases during the beer fermentation. Some other acids are produced exclusively as a consequence of yeast metabolism. They can directly affect the flavour of beer by decreasing its pH level.

h) Fatty acids are low-grade components of wort but can increase in concentration during fermentation and maturation. They lead to goaty, foamy, or fatty flavours and are renowned as ordinary flavour features in both lagers and ales; however, they are well established in lagers because of the predisposition of some lager yeast strains to generate greater quantities of fatty acids than do strains of ale yeast.

i) Yeast also produces some nitrogen compounds during fermentation and maturation such as amino acids and lower peptides, with this contributing to shape the flavour and provides an increase in palate roundness. For that reason, collecting the yeast too soon can yield empty, dry beers even if lagered afterwards for a extended period.

The right selection of yeast with the basic brewing characteristics needed is essential from a product quality and economic point of view. The criteria for yeast selection can be different in accordance to the requirements of the brewing facilities and the beer style, but they are expected to incorporate the following aspects:

- fast fermentation;
- yeast stress tolerance;
- flocculation;

- rate of decrease at the temperature wanted;
- flavour of the final product;
- superior yeast storage features;
- stability against mutation; and
- stability against degeneration.

Fermentation time/temperature profiles vary extensively throughout the industry for beer lagers. Conventional lager brewing comprises pitching the yeast between 5 and 6°C letting the temperature to increase between 8 and 9°C. This usually results in a beer with improved quality since low fermentation temperature slows down the development of by-products, esters, fusel alcohols, and diacetyl, all of which can be inappropriate in lagers (as mentioned before). On the other hand, the lag period is normally longer at lower fermentation temperatures. At the end of primary fermentation the temperature can be reduced by one to one and a half degrees Celsius per day and relocated to a lager cellar where kept between 4 and 5°C. Although frequently the yeast is pitched between 7 and 8°C and after a few days the temperature is increased to 10 or 11°C. Other breweries tend to use the same initial temperature but then increased to 14 and 15°C. A number of brewers are acknowledged to pitch between 12 and 14°C and then raise the temperature to 18°C.

Microorganisms causing spoilage during brewing and beer processing are limited to a few varieties of bacteria, wild yeasts, and molds. This is thanks to the beer being a somewhat hostile growth medium for most beer spoilage microorganisms. The alcohol content, low pH, and the presence of hop constituents are inhibitory, while the lack of nutrients limits the development of those cells that do survive. Nevertheless, these can interfere with the fermentation itself or have damaging effects on beer flavour and shelf life.

1.3 OPTIMAL CONTROL AND OPTIMISATION ALGORITHMS

The ancient Egyptians were the first to brew beer. The earliest true large-scale breweries date from the early 1700s when wooden vats of 1500 barrels capacity were introduced. Even some process control was attempted in these early breweries, as indicated by the recorded use of thermometers in 1757 and the development of primitive heat exchangers in 1801. During the late 1800s Hansen started his pioneering work at the Carlsberg brewery and developed methods for isolating and propagating single yeast cells to produce pure cultures and established sophisticated techniques for the production of starter cultures (Seborg et al, 1989).

Biotechnological processes, as the fermentation one, may be conveniently classified according to the mode chosen for process operation: either batch, fed-batch or continuous (Johnson, 1987). During batch operation of a process, no substrate is added to the initial charge nor is product removed until the end of the process. Nevertheless, continuous operation is more economic, where substrate is continually added and product continually removed. Fed-batch processes introduce the greatest challenge since the feed rate may be changed during the process but no product is removed until the end.

The heuristic method of trial and error, which is used to find an optimal or pseudo-optimal operating regime by manipulating the process technological parameters, is one of the oldest optimisation methods. According to Stengel (1994), there are in general three reasons for process control: to ensure or enhance process stability, to suppress the influence of disturbances and finally to optimise the process performance. The formulation of an optimal control problem requires the following components (Stanbury et al, 1995):

i) A model of the system to be controlled: this is the constitutive equations, together where applicable with end-state conditions and response transformation. It characterises the system and enables the effect of all iterative controls on the system to be predicted.

ii) The constraints upon the design: they limit the range of permissible solutions and fix many systems properties.

iii) The demands presented to the system as a design goal (objective, criterion or index): is derived from a design value statement. The problem is to decide the control that gives the least or greatest value of this index.

Many control design problems are based on two phases: choosing a control structure and choosing an optimal set of parameters given the control structure, although a design process may pass repeatedly through these two phases (Queeinec et al, 1992). The parameters are chosen to satisfy a set of inequalities specifying design objectives or to minimise a criterion subject to those inequalities.

The control of a fermentation process is based on the measurement of physical, chemical or biochemical properties of the fermentation broth and the manipulation of physical and chemical environmental parameters such as temperature, dissolved oxygen tension and nutrient concentrations (Omstead, 1990). The microorganisms or biomass concentrations are the central feature of fermentation affecting the rates of growth, substrate consumption and product formation.

Czyzyk et al (1997) state that numerical optimisation methods can be divided in two major sub fields according to the problem they addressed: Continuous (Unconstrained or Constrained) and Discrete.

- Continuous unconstrained optimisation techniques
    i. Non-linear equations: Systems of non-linear equations come up as constraints in optimisation problems, but also arise, when differential and integral equations are discretized. Newton's method, modified and enhanced, forms the basis for most of the software used to solve systems of non-linear equations. Nearly all the computational cost of Newton's method is linked with two operations: evaluation of the function and the Jacobian matrix, and the solution of the linear system. Convergence of Newton's method can be assured if the initiation is sufficiently close to the solution and the Jacobian at the solution is non-singular. The

following known commercial software attempts to overcome these two disadvantages of Newton's method by allowing approximations to be used in place of the exact Jacobian matrix and by using two basic strategies-trust region and line search-to improve global convergence behaviour (More and Wright, 1993): GAUSS, IMSL, LANCELOT, MATLAB, MINPACK-1, NAG (in FORTRAN), NAG (in C), NITSOL, and OPTIMA. Other methods used to solve these kind of problems include: Trust Region and Line-search Methods, Truncated Newton Method, Broyden's Method, Tensor Methods and Homotopy Methods.

ii. Non-linear Least Squares: Least squares problems often take place in data-fitting applications. From an algorithmic point of view, the feature that characterizes least squares problems from the general unconstrained optimisation problem is the structure of the Hessian matrix. Some methods used for solving this kind of problems are the Gauss-Newton Method, Levenberg-Marquardt Method, Hybrid Methods and Large Scale Methods.

iii. Global Optimisation: One of the greatest complications in Non-linear Programming is that some problems show what is called "local optima"; that is, false solutions that merely satisfy the requirements on the derivatives of the functions. Algorithms that propose to overcome this difficulty have been branded Global Optimisation. Techniques for solving this kind of problems include: Dynamic Programming, Branch and Bound (Mixed Integer Programming, Constraint Satisfaction Techniques, DC-Methods, Interval Methods and Stochastic Methods), Simulated Annealing, Genetic Algorithms, Other Stochastic Methods, Continuation Methods and Other Heuristics. Additional information on the Genetic Algorithms technique can be found in Chapter 4.

- Continuous constrained optimisation techniques
  i. Linear Programming: The basic problem in linear programming is to minimise a linear objective function of continuous real variables, subject to linear constraints. Software for linear programming (including

network linear programming) commonly requires more computer cycles than software for all other kinds of optimisation problems combined. The simplex algorithm, named like that because of the geometry of the feasible set, motivates the extensive majority of available software packages for linear programming. However, this situation can change in the future, as more software for interior-point algorithms becomes available. However, since most models of fermentation processes are of a non-linear nature, the use of these techniques for process optimisation purposes is not very common.

ii. Non-linear Constrained Optimisation: The general constrained optimisation problem is to minimise a non-linear function subject to non-linear constraints. The main techniques that have been proposed for solving constrained optimisation problems are reduced-gradient methods, sequential linear and quadratic programming methods, and methods based on augmented Lagrangians and exact penalty functions. One of these methods, Sequential Quadratic Programming is the main focus of research in Chapter 5.

iii. Bound Constrained Optimisation: Bound-constrained optimisation problems play an important role in the development of software for the general constrained problem because many constrained codes reduce the solution of the general problem to the solution of a sequence of bound-constrained problems. It is also important in applications because parameters that describe physical quantities are often constrained to lie in a given range. Newton Methods and Gradient-Projection Methods are the basic tools for this sort of optimisation problems. The Gradient Method of Function Space and the DISOPE algorithm (Dynamic Integrated System Optimisation and Parameter Estimation) employ this technique in Chapter 3.

iv. Network Programming: As the name designates, network problems arise in applications that can be represented as the flow of products in a network. Herewith, the resulting programs can be linear or non-linear and

can cover a large number of applications such as: Transportation Problems, Assignment Problems, Maximum Value Flow, Shortest Path Problem and Minimum Cost Flow Problem. Commercial software such as NETFLOW and RELAX-IV deal with this kind of problems (More and Wright, 1993).

- Discrete optimisation techniques

  i. Stochastic Programming: For many actual problems, the problem data cannot be identified precisely for at least two reasons: the first reason is due to simple measurement error, the second and more fundamental reason is that some data represent information about the future and simply cannot be known as a fact. Thus, the solutions obtained may perhaps be optimal for the specific problem but may not be optimal for the situation that actually occurs. Being able to take uncertainty into account is critical for many problems where the essence of the problem is dealing with the doubt in some optimal path. Stochastic programming enables the modeller to create a solution that is optimal over a set of scenarios. MSLIP (More and Wright, 1993), for example, is a software technique that uses a nested Bender's decomposition approach to solve multistage problems.

  ii. Integer Programming: In many applications, the solution of an optimisation problem only makes sense if the variables are integers. Integer programming problems, such as the fixed-charge network flow problem and the famous travelling salesman problem are frequently expressed in terms of binary variables. Although a number of algorithms have been proposed for the integer linear programming problem, the *branch and bound* technique is used in almost all of the software nowadays. This technique has demonstrated to be reasonably efficient on practical problems, and it has the additional advantage that it solves continuous linear programs as sub problems, that is, linear programming problems without integer restrictions. The CPLEX, FortLP, LAMPS, LINDO, MIP III, OSL, and PC-PROG software packages use the branch

and bound technique to solve mixed-integer linear programs (More and Wright, 1993).

With regard to fermentation, dynamic optimisation of batch processes attempts to find the best initial profiles during a batch run. The methods for this optimisation can be classified in three categories (Bonvin, 1997):

i. One time optimisation: an optimal control problem is formulated based on a dynamic model of the process. The solution provides the required input trajectories.
ii. Batch to batch optimisation: the additional information available with the completion of each batch run is used to improve future operations. The calculations required by these methods are normally carried out in the intermediate period between two consecutive batch runs.
iii. Online optimisation: these methods try to compensate for the presence of modelling errors and disturbances when input profiles computed offline become sub-optimal. It is accomplished by repeating online model-based optimisation accompanied by system identification several times during a batch run using real time measurements, introducing feedback in the calculation of the input profiles.

1.4 SUMMARY

The basics concepts of beer fermentation for industrial processing have been reviewed in this chapter. An initial introduction on related terms for fermentation and beer processing is included. The history of beer control have been also integrated, trying to notice how fermentation has changed with time. A brief description on different ways for beer processing has been discussed; including the different kinds of beers according to the preliminary constituents used and the attributes of the by-products present at the end of the fermentation.

Optimal control techniques and a general description of the known optimisation methods are also subject of attention in this chapter. After that, old and new optimisation algorithms (and some software packages used at the moment) have been presented with some information on the basics of every technique.

CHAPTER II

MODELLING AND SIMULATION OF BEER FERMENTATION PROCESSES

The modelling of fermentation processes is a basic part of any research in fermentation process control. Since all the optimisation work to be done is based on the reliability of the model equations, they are important for the right design. Alcoholic brewery fermentation is the main objective of this work.

2.1 MODELLING OF FERMENTATION PROCESSES

In fermentation, an accurate mathematical model is a prerequisite for the control, optimisation and the simulation of a process. Models used for on-line control and those used for simulation will not generally be the same, even if they pertain to the same process, because they are used for different purposes. In a quite general approach to modelling, a priori knowledge is the basis for a set of mathematical equations with unknown parameters (Roux et al, 1996). Estimating algorithms, if properly chosen, yields the parameter values after processing of data coming from measurements on the system. Validation as a continuing exercise could develop the best model equations.

An investigation into causes of the problems, associated with a system-theoretic approach to control of fermentation, has shown that it is not yet clear which mathematical framework is best fitted for modelling. Generally, in batch or fed batch fermentation processes there is no steady state. Growth and product formation rates vary with time due to a dependence on the present state of the batch as characterised by biomass, substrate and product concentrations, dissolved oxygen tension, nutrient feed rates and also on the condition of the culture (Johnson, 1987). These equations are generally non-linear.

The formulation of mathematical fermentation process models, from the perspective of system analysis, is usually realised in three stages:

i. Qualitative analysis of the structure of a system, usually based on the knowledge of metabolic pathways and biogenesis of the desired product,
ii. Formulation of the model in a general mathematical form. This stage is sometimes called the structure synthesis of the process functional operator;
iii. Identification and determination of numerical values of model constants and/or parameters, which is based on experimental or other operating data from a real process.

The process of creating the mathematical model of fermentation starts usually from a simplified scheme of reactions derived from knowledge of metabolic pathways involved. Each metabolic reaction step is characterised by the reaction stoichiometry on one hand and by the flux (represented by the reaction velocity or rate) on the other. Reactions are usually approximated by using one of the relationships derived from the theory of enzymatic or chemical reactions. The most frequent relationships employed suitable for this purpose are summarised by Volesky and Votruba (1992) as in the following equations. Table 2.1 shows the description of the parameters used in these equations.

Parameter	Description
k	Rate of change of the phenomenon
K	Active site
n	Natural number
r	Relationship being considered
S	Substrate concentration

Table 2.1 Nomenclature used for the different relationships

$r_1 = kS$	Linear relationship between the rate of the phenomenon and the reaction Substrate concentration.
$r_2 = kS^n$	Derived from the Freundlich absorption isotherm, characteristic for most hydrolytic reactions.
$r_3 = \frac{kS}{K+S}$	Typical relationships used in fermentation, represents the rate of change of a phenomenon controlled by chemisorption of the Substrate onto one active site such as the molecule of an enzyme.
$r_4 = \frac{kS^n}{K+S^n}$	Modification of the preceding case where more than one active site is present on each bio-catalytic molecule.
$r_5 = k\left[1 - \exp^{(-S/K)}\right]$	Not usual type of rate relationship recommended for describing the process dynamics, based on a purely physical interpretation derived from equations for movement of a mass point in an environment characterised by dissipation forces.
$r_6 = k\exp^{(-S/K)}$	The substance with concentration S is considered as directly participating in the dissipation of kinetic energy during the course of the reaction.

$$r_7 = \frac{kK}{K + S}$$

Based on the principle of hypothetical reversible blocking of the active reaction site by chemisorption of a substance with concentration S.

$$r_8 = \frac{kK}{K + S^n}$$

Derived from inhibition of a larger number of active reaction sites of a certain biochemical process bottleneck.

Frequently, the use of these rate relationships is made by combination between them based on superimposition of several phenomena in the given sub-system. Sometimes this aspect becomes relevant only when it comes to model identification. This is usually accomplished either by plotting the derived numerical relationships for rates against concentrations of the substrate or of the product. The plot and correlation of the rates against each other enables estimation of local yield coefficients or eventually of their mutual relationships.

The determination of numerical values of the mathematical model parameters is based on an appropriate method as an important part of modelling. According to Volesky and Votruba (1992); these methods can be divided into:

i) linear and non-linear regression: based on conventional methods of mathematical statistics,
ii) momentum analysis of experimental data: using techniques derived from momentum analysis for expressing numerical values of model parameters and
iii) adaptive identification and estimation of model parameters: in which the computer continuously re-evaluates model parameters so that the behaviour of the process can be predicted and controlled.

In the simulation application, model equations are solved for different initial and boundary conditions according to a certain scenario based on the planning of

simulation experiments. Simulation studies make possible testing of novel or alternative technological variants of the process such as the change from a batch to continuous-flow cultivation, or to the use of an immobilised-cell technology.

The understanding and study of any process, requires a mathematical representation or model of the process. The process may have an input-output representation or a time series. The model is based on the prior physical or subjective knowledge about the process itself, the measured data on the inputs and the outputs, and the physical and engineering laws governing the working of the process.

If the model is a complete and exact representation of the process, it is called a deterministic model, and the process is then called a deterministic process. The parameters of such a model are precisely known, and the model can be used to produce exact prediction of the process response from the past data. Nevertheless, most real life processes cannot be represented by this kind of model, because of the dynamic nature of the process and the lack of information and other uncertainties being associated with the available data. A model that incorporates noise or disturbance terms to account for such imprecision in the knowledge of the process is in that case called a stochastic model.

The design of a control system is usually based upon a linear model of the plant to be controlled, for the good reason that the assumption of linearity makes the dynamical behaviour much easier to analyse. In practice, however, all systems are usually non-linear and, therefore, may exhibit forms of behaviour that are not at all apparent from the study of the linearised versions. The model is not expected to be a reconstruction of the process, rather it is intended to serve as a set of operators on the identified set of inputs, producing similar output as expected from the process. The problem is that in real life the process output is usually contaminated with noise and other disturbances, whereas ideally the model should follow the true output of the underlying representative process, which is unknown. There can be different models for the same process, although no model can be said to be the best.

According to previous research about fermentation processes, three different modes can be distinguished (Ferreira, 1997):

1. *Batch fermentation* refers to a partially closed system in which most of the materials required are loaded onto the fermentor, decontaminated before the process starts and then removed at the end. Conditions are continuously changing with time, and the fermentor is an unsteady-state system, although in a well-mixed reactor, conditions can be assumed to be consistent throughout the reactor at any instant of time.

2. *Continuous culture* is a technique involving feeding the microorganism used for the fermentation with fresh nutrients and, at the same time, removing spent medium plus cells from the system. A time-independent steady state can be attained which enables one to determine the relations between microbial behaviour and the environmental conditions.

3. *Fed-batch processes* are commonly used in industrial fermentation. Fed-batch fermentation is a production technique in between batch and continuous fermentation. They improve control possibilities, such as computer based fermentation systems. A fed-batch is useful in achieving high concentrations of product because of high concentrations of cells for a relative large period of time.

Fermentation processes are used for producing many fine substances such as amino acids, antibiotics, baker's yeast, enzymes, etc. Among the modes of operation (batch, fed-batch and continuous), the fed-batch technique is often used in industry due to its ability to overcome the catabolic repression or glucose effect, which usually occurs during production of these fine chemicals (Vanichsriratana et al, 1997).

Fed-batch fermentation can be the best option for some systems in which the nutrients or any other substrates are only sparingly soluble or are too toxic for adding the whole requirement for a batch process at the start. In the fixed volume fed-batch

process, the limiting substrate is fed without diluting the culture. The culture volume can also be maintained practically constant by feeding the growth limiting substrate in undiluted form.

Two cases can be considered in fed-batch fermentation: the production of a growth associated product and the production of a non-growth-associated product. In the first case, it is desirable to extend the growth phase as much as possible, minimising the changes in the fermentor as far as specific growth rate, production of interest and avoiding the production of by-products. For non-growth associated products, the fed-batch would have two phases: a growth phase, in which the cells are grown to the required concentration, and then a production phase, in which carbon source and other requirements for production are fed to the fermentor.

A variable fed-batch is one in which the volume changes with the fermentation time due to the substrate feed. The way this volume changes is dependent on the requirements, limitations and objectives of the operator. A proper feed rate, with the right component constitution, is required during the process. The production of by-products, which are generally related to the presence of high concentrations of substrate, can also be avoided by limiting its quantity to the amounts that are required solely for the production of the biochemical. When high concentrations of substrate are present, the cells get overloaded, in that, the oxidative capacity of the cells is exceeded, and due to the Crabtree effect (Reynders et al, 1997) products other than the one of interest are produced, reducing the efficacy of the carbon flux. Moreover, these by-products have shown to even contaminate the product of interest, such as ethanol production in baker's yeast production, and to impair the cell growth reducing the fermentation time and its related productivity.

The optimal strategy for the fed-batch fermentation of most organisms is to feed the growth-limiting substrate at the same rate that the organism utilises the substrate: that is, to match the feed rate with demand for the substrate. Regardless of the type of control, both mathematical model availability and measurement possibilities influence the design. The subsequent mathematical model has the following assumptions: the feed is provided at a constant rate, the production of mass of

biomass per mass of substrate is constant during the fermentation time; and a very concentrated feed is being provided to the fermentor.

Table 2.2 shows the description of the parameters used for the fed-batch fermentation equations:

Parameter	Description
alpha	Constant yield
beta	Constant yield
F	Substrate feed rate
G	Constant value defined by experimental data
K_d	Specific death rate
K_i	Inhibition constant
K_m	Inhibition constant
P	Product concentration
P_m	Maximum product concentration
q_p	Specific production rate of product
R	Constant value defined by experimental data
r_p	Product formation rate
S	Substrate concentration in the fermentor
S_0	Substrate concentration in the feed
t	Time
T	Constant value defined by experimental data
u	Specific growth rate
u_{max}	Maximum growth rate
V	Volume of the fermentor
X	Biomass concentration
X_0	Biomass at the beginning of the fermentation
Y_0	Initial value of the yield factor
$Y_{x/s}$	Yield factor

Table 2.2 Description of the parameters used in fed-batch fermentation

Thus, the parameter equations related to fed-batch fermentation are:

$$u = \frac{(FY_{x/s})}{X}$$ Specific Growth Rate

$$X_t = X_0 + FY_{x/s}t$$ Biomass (as a function of time)

$$P = P_i + q_p X_0 t + \frac{q_p FY_{x/s}t^2}{2}$$ Product Concentration (non-growth associated)

$$P = P_i + r_p t$$ Product Concentration (growth associated)

In variable fed-batch fermentation, an additional element should be considered: the feed. Consequently, the volume of the medium in the fermentor varies because there is an inflow and no outflow. For the next mathematical development, the assumptions are: specific growth rate is uniquely dependent on the concentrations of the limiting substrate; the concentration of the limiting substrate in the feed is constant; the feed is sterile; and the yields are constant during the fermentation time.

Accordingly, the differential equations for the components for fed-batch fermentation are:

$$F = \frac{dV}{dt}$$ Overall component

$$\frac{dX}{dt} = \frac{X(uV - K_d V - F)}{V}$$ Biomass

$$\frac{dS}{dt} = \frac{F(S_0 - S)}{V} - \frac{uX}{Y_{x/s}}$$ Substrate

$$\frac{dP}{dt} = q_p X - \frac{PF}{V}$$ Product

A list of growth models that can be found in bio-transformations is included in Table 2.3 (Agrawal et al, 1989):

Model	**Form**
Monod	$u = \frac{u_{max} S}{K_m + S}$
Constant yield	$Y_{x/s} = Y_0$
Substrate inhibition	$u = \frac{u_{max} S}{K_m + S + \frac{S^2}{K_i}}$
Constant yield	$Y_{x/s} = Y_0$
Substrate inhibition	$u = \frac{u_{max} S(1 - TS)}{K_m + S + \frac{S^2}{K_i}}$
Variable yield	$Y_{x/s} = \frac{Y_0 (1 - TS)}{1 + RS + GS^2}$
Substrate and product inhibition	$u = \frac{u_{max} S}{K_m + S + \frac{S^2}{K_i}}$
Inhibitions	$u = u_{max}\left(1 - \frac{P}{P_m}\right)$
	$q_p = alpha \cdot u + beta$
Constant yields	alpha, beta and $Y_{x/s}$

Table 2.3 Growth models found in biotransformations

One of the most used approach for process optimisation of fed-batch fermentation is to calculate an optimal feed-rate profile, which will optimise a given objective function. Temperature, speed of agitation, flow rates, pH, dissolved oxygen, redox potential of broth are examples of other commonly controlled variables; these sensors usually respond quickly enough and so, proportional feedback controllers are often suitable for these variables. The choice of each parameter is system dependent

and the decision should be based on ease and experimental data. Given that many state variables in fermentation processes, such as biomass, substrate and product concentration, are difficult to measure on-line, many methods have therefore been developed for on-line estimation of these state variables.

Substrate, on the other hand, is a particularly important parameter to control due to eventual associated growth inhibitions and to increase the effectiveness of the carbon flux, by reducing the amount of by-products formed and the amount of carbon dioxide evolved. The production of these by-products is undesirable because it reduces the efficacy of the carbon flux in fermentation (see Chapter 1 for more information). The production of these components takes place whenever the substrate is provided in quantities that exceed the oxidative capacity of the cells. This approach has been used in the fermentation of *Saccharomyces cerevisiae*, in which acid production rate is used to provide on-line estimates of the specific growth rate (Leigh, 1986).

The feeding mode influences fed-batch fermentation by defining the growth rate of the micro-organisms, the effectiveness of the carbon cycle for product and minimisation of by-product formation. Inherently related with the concept of fed-batch, the feeding mode allows many variances in substrate or other components constitution and provision modes and consequently, better controls over inhibitory effects of the substrate and/or product.

Overall, the control of a fed-batch fermentation process can implicate many difficulties (Ferreira, 1999): low accuracy of on-line measurements of substrate concentrations, limited validity of the feed schedule under a variety of conditions and prediction of variations due to strain modification or change in the quality of the nutrient medium. These aspects indicate the need of a fed-batch fermentation plan which is model independent, identifies the optimal state on-line, includes negative feedback control into the nutrient feeding system and considers a saturation kinetic model, a variable yield model, variation in feed substrate concentration and product inhibited fermentation.

In an open-loop operation system, a predetermined feed schedule is used (Agrawal et al, 1989). This approach considers that the system can be exactly converted into a set of mass balance equations that contains the specific growth rates. Yet, it is easy to presume that due to a non-identified physiological problem of the cells, the specific growth rate can be either higher or lower than the one that was previously established. The open-loop feed policy, as a result, does not always result in an optimal operation.

The feedback control algorithm requires a reliable on-line estimate of the specific growth rate. Since the objective of the algorithm is to optimise the cell-mass production by controlling the specific growth rate (u) at an optimum value u_{opt}, a feedback law can be defined as below (Ferreira, 1999). This relation can be used to manipulate the feed flow rate (F_{in}) to the fermentor in a specific time (t_n), where K_c is a controller constant which is assumed to be positive. When u is different from u_{opt}, either $S < S_{opt}$ or $S > S_{opt}$. Then, the positive sign in the equation applies to the first case and, similarly, the negative sign applies to the second case:

$$F_{in}(t_{n+1}) = F_{in}(t_n) \pm K_c \left[u_{opt}(t_n) - u(t_n) \right] \qquad \text{Feedback law}$$

The use of fed-batch culture by the fermentation industry takes advantage of the fact that the concentration of the limiting substrate may be maintained at a very low level, in consequence: avoiding repressive effects of high substrate concentration, controlling the organism's growth rate and as a result controlling the oxygen demand of the fermentation (McNeil and Harvey, 1990 & Neway, 1989).

The degrees of freedom for the determination of the optimum conditions in batch processes, maximum product with minimum cost and time, are often a combination of the initial conditions, the set-point profile and the time allowed for the transformation phase. The procedure used for determining an acceptable set-point profile is called dynamic optimisation, but the profile obtained with this method will only be optimal for the specific model and parameter values used in the optimisation (Becerra and Roberts, 1998a).

2.2 A BEER FERMENTATION MODEL

The mathematical model chosen to be part of the simulation (and subsequently the optimisation process) is the kinetic model by Andres-Toro et al (1998) since it has been obtained from many experimental studies at laboratory scale showing good results. It takes into account realistic aspects of the process such as characteristics of the wort and yeast, and also two important by-products of the fermentation: ethyl acetate and diacetyl (see Chapter 1 for more information on by-products effect).

A batch fermentation process such as on the beer fermentation process is complex because of the biological phenomena taking place and the dynamic nature of the process itself (Andres-Toro et al, 1997b). When formulating a model of a microbial process, feasibility is a guiding principle. A very frequent mistake committed is the creation of a very complex model including different approaches available in the literature, disregarding their relevance to the overall goal, which should always be the simplest and yet adequately accurate way of describing the real process that would enable its simulation by calculations. Such a model can then be conveniently used for the prediction of optimal operating conditions of a technological process.

With data obtained experimenting in the laboratory by Andres-Toro et al, (1998) developed a new model of the fermentation dynamic behaviour based on the activity of suspended biomass. The model was obtained from many experimental studies at laboratory scale with the necessary equipment as can be seen in Figure 2.1. The model has shown good results, and it should be noted that it takes into account realistic aspects of the process, such as the characteristics of wort and yeast.

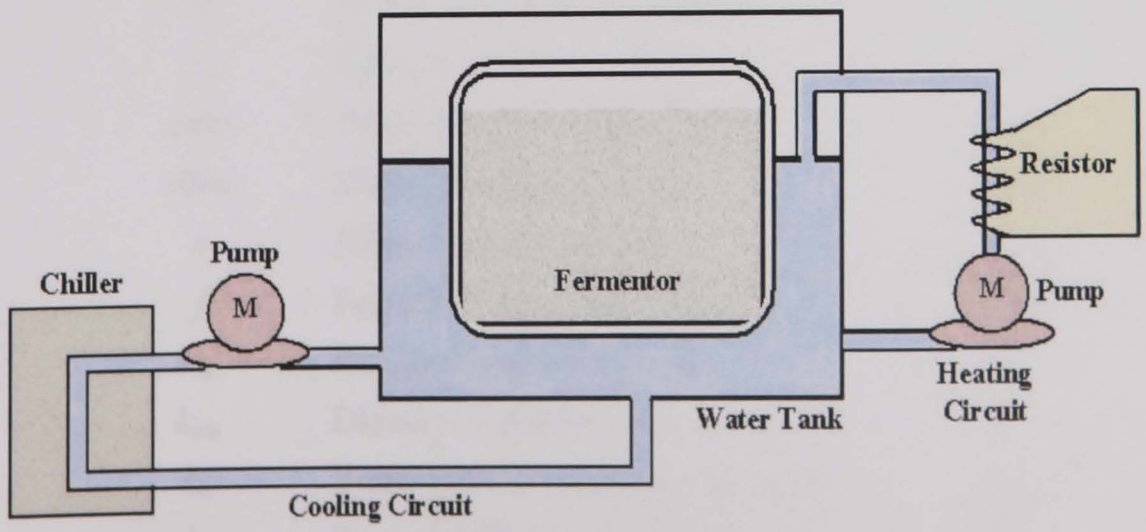

Figure 2.1 Experimental set-up (Andres-Toro et al, 1998)

Thus, some equations of the model are devoted to the biomass behaviour: part of it settles slowly and is inactive, while the active biomass awakes from latency to start growing and producing ethanol, etc. An important effect of the temperature over the process acceleration was recorded. A scheme of the process with its main variables is presented in Figure 2.2. Table 2.4 describes the nomenclature used for this mathematical model.

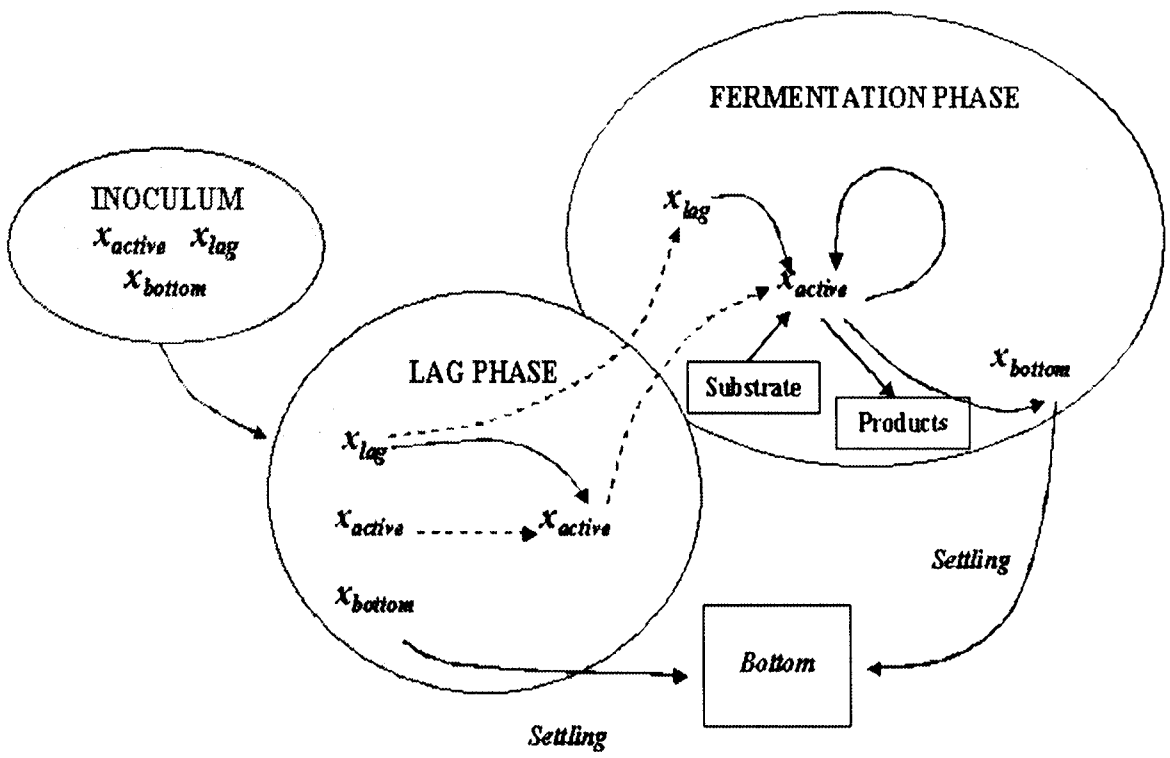

Figure 2.2 Process scheme for the kinetic model (Andres-Toro et al, 1998)

Parameter	Description	Unit
μ_a	Ethanol production rate	h^{-1}
μ_D	Specific yeast settling down rate	g/l
μ_{eas}	Ethyl acetate coefficient rate	g/l
μ_{lag}	Specific rate of latent formation	h^{-1}
μ_s	Substrate consumption rate	h^{-1}
μ_x	Specific yeast growth rate	h^{-1}
acet	Ethyl acetate concentration	ppm
diac	Diacetyl concentration	ppm
e	Ethanol concentration	g/l
f	Fermentation inhibitor factor	g/l
k_{dc}	Diacetyl appearance rate	h^{-1}
k_{dm}	Diacetyl reduction rate	h^{-1}
k_m	Yeast growth inhibition parameter	g/l
k_s	Sugar inhibition parameter	g/l

s	Concentration of sugar	g/l
s_0	Initial concentration of sugar	g/l
t	Time	h
T	Temperature	°C
x_{active}	Suspended active biomass	g/l
x_{dead}	Suspended dead biomass	g/l
x_{lag}	Suspended latent biomass	g/l

Table 2.4 Nomenclature in the mathematical model

Biomass is segregated into three different types of cells: lag, active and dead. The whole process can be divided in two successive phases: a lag phase and a fermentation phase. Herewith, the enunciation of the model is as follows:

Lag Phase
$$\frac{dx_{lag}}{dt} = -\mu_{lag} \cdot x_{lag} \tag{2.1}$$

Fermentation Phase
$$\frac{dx_{active}}{dt} = \mu_x \cdot x_{active} - k_m \cdot x_{active} + \mu_{lag} \cdot x_{lag} \tag{2.2}$$

$$\frac{dx_{dead}}{dt} = k_m \cdot x_{active} - \mu_D \cdot x_{dead} \tag{2.3}$$

$$\frac{ds}{dt} = -\mu_s \cdot x_{active} \tag{2.4}$$

$$\frac{de}{dt} = \mu_a \cdot f \cdot x_{active} \tag{2.5}$$

To describe the evolution of the by-products that have an important impact (ethyl acetate contributes with a fruity odour and flavour, and diacetyl makes beer heavy and sweet flavoured), the following equations are established:

$$\frac{d(acet)}{dt} = \mu_{eas} \cdot \mu_s \cdot x_{active} \tag{2.6}$$

$$\frac{d(diac)}{dt} = k_{dc} \cdot s \cdot x_{active} - k_{dm} \cdot diac \cdot e \tag{2.7}$$

The remaining parameter equations can be calculated as follows:

$$x_{total} = x_{active} + x_{lag} + x_{dead} \tag{2.8}$$

$$\mu_x = \frac{\mu_{x0} \cdot s}{0.5 \cdot s_0 + e} \tag{2.9}$$

$$\mu_D = \frac{0.5 \cdot s_0 \cdot \mu_{D0}}{0.5 \cdot s_0 + e} \tag{2.10}$$

$$\mu_s = \frac{\mu_{s0} \cdot s}{k_s + s} \tag{2.11}$$

$$\mu_a = \frac{\mu_{a0} \cdot s}{k_s + s} \tag{2.12}$$

$$f = 1 - \frac{e}{0.5 \cdot s_0} \tag{2.13}$$

Since the process depends on temperature, the value of all parameters of the model is calculated by exponential equations of the Arrhenius type ($\mu = A \cdot e^{\frac{B}{RT}}$) as follows:

$$\mu_{x0} = 1.095 \times 10^{47} \cdot e^{\frac{-63720}{1.99536 \cdot (T+273.15)}} \tag{2.14}$$

$$k_m = 3.373 \times 10^{56} \cdot e^{\frac{-76450}{1.99536 \cdot (T+273.15)}} \tag{2.15}$$

$$\mu_{eas} = 1.129 \times 10^{39} \cdot e^{\frac{-53056}{1.99536 \cdot (T+273.15)}} \tag{2.16}$$

$$\mu_{D0} = 4.889 \times 10^{14} \cdot e^{\frac{-20020}{1.99536 \cdot (T+273.15)}} \tag{2.17}$$

$$\mu_{s0} = 6.232 \times 10^{-19} \cdot e^{\frac{23254}{1.99536 \cdot (T+273.15)}} \tag{2.18}$$

$$\mu_{a0} = 26.3865 \cdot e^{\frac{-2528.6}{1.99536 \cdot (T+273.15)}} \tag{2.19}$$

$$\mu_{lag} = 2.2041 \times 10^{13} \cdot e^{\frac{-18959}{1.99536 \cdot (T+273.15)}} \tag{2.20}$$

$$k_s = 1.1081 \times 10^{-52} \cdot e^{\frac{68249.2}{1.99536 \cdot (T+273.15)}} \tag{2.21}$$

The non-linearity of the model can be observed particularly with these Arrhenius functions of temperature that include exponential values within.

The diacetyl formation and reduction constants have been calculated and simplified for different temperature values by non-linear approximations from previous work by Garcia et al (1994); and are included in the model as follows:

$$k_{dc} = 0.000127672 \tag{2.22}$$

$$k_{dm} = 0.00113864 \tag{2.23}$$

The objective function defined intends to reach the required ethanol level in the industrial fermentation without quality loss or contamination risks and also maintaining a good implementable profile for the industry. The following sub functions are defined including penalty parameters to obtain an approximation of an objective function to be maximised:

$$J_1 = +10 \cdot e_{end} \tag{2.24}$$

Measures and weights up the final ethanol growth to be used.

$$J_2 = -5.73 \times 10^{-8} \cdot e^{(95 \cdot diac_{end})} \tag{2.25}$$

Limits the diacetyl concentration at the end to be less than 0.2 ppm.

$$J_3 = -1.16 \times 10^{-29} \cdot e^{(4.6 \cdot acet_{end})} \tag{2.26}$$

Limits the level of ethyl acetate at the end to be less than 15 ppm.

$$J_4 = -\int_0^{t_f} 9.91 \times 10^{-7} \cdot e^{(2.31 \cdot T)} dt \tag{2.27}$$

Limits the temperature under 16°C along the entire process to avoid the spoilage risk by *Lactobacillus Plantarum*.

$$J_5 = -\sum_{i=1}^{160} \frac{|T_{i+1} - T_i|}{\Delta t} \tag{2.28}$$

Penalises brisk changes in temperature for every step along the whole process, thus obtaining a smoother profile.

These terms combine to obtain an overall cost function of the process:

$$J = J_1 + J_2 + J_3 + J_4 + J_5 \qquad (2.29)$$

A temperature profile that maximises this function in a fixed time is required. As an initial reference, the same temperature profile used by the industry, according to Andres-Toro et al, (1997b), was taken for a solution along 200 hours (see Figure 2.4). It is significant to note that some parameters and equations have been taken from the original research literature and some others have been slightly modified in order to follow the industrial patterns in relation to the beer fermentation behaviour of the byproducts. In particular, the diacetyl and ethyl acetate parameters (equations 2.22 and 2.23) have been obtained, as previously mentioned, from work by Garcia et al (1994), instead of a set of polynomial equations as it can be found on the original work by Andres-Toro et al, (1998). The sub functions J_2 and J_3 (corresponding to equations 2.25 and 2.26) have also been adjusted in order to appropriately limit the final concentration of these byproducts within a specific range of 0.2 and 15 ppm respectively used in practice by the beer industry, as it was found in the literature by Gee and Ramirez (1994) and also specified for beer production according to other sources (Kunze, 1996).

The use of this objective function grouping all the aspects of the problem together makes easier the application of an optimisation control technique since it can be used as the fitting function.

In addition to achieve a maximum for the objective function described, the process can also be accelerated. This could be done by including another objective function to the process that minimises the total fermentation time. Taking this into consideration, the task becomes a multi-objective optimisation problem that can also be solved by optimal control techniques, however, this approach is beyond the aim of this research.

The mathematical model has taken basic principles of beer fermentation from work by Gee and Ramirez (1994) on the development of a new representation of beer processes based upon known biochemical pathways, the main aim of this research being to focus in on-line optimisation, control studies and process optimisation purposes. Other research work by Garcia et al (1994) has brought into consideration the ability to predict some of the by-product's concentration by means of kinetic equations. More recently, Andres-Toro et al (1999) has used this model yet again to illustrate the benefit of a genetic optimisation technique developed for the optimisation of non-linear processes.

2.3 COMPUTER SIMULATION OF THE SELECTED FERMENTATION PROCESS

SIMULINK and MATLAB have been chosen to simulate the selected mathematical model of the beer fermentation process. All the differential equations of the mathematical model have been integrated in a MATLAB file (m-file) and included as an S-function block in SIMULINK. The initial parameters and all the other required functions have also been included in the S-function script.

The temperature is the only input of the model and is included in SIMULINK by means of a MATLAB two dimensions linear look-up table that allows x (time in hours) and y (temperature in degrees centigrade) vectors to be included together. Both interpolation between coordinate points and/or a simple set of steps profile can be used with this look-up table. The value of the model equation variables is extracted from the S-function with a Demux block that splits the output signal into the seven vectors that correspond to the differential equations variables of the mathematical beer process. These output vectors can then be analysed and its behaviour monitored and controlled.

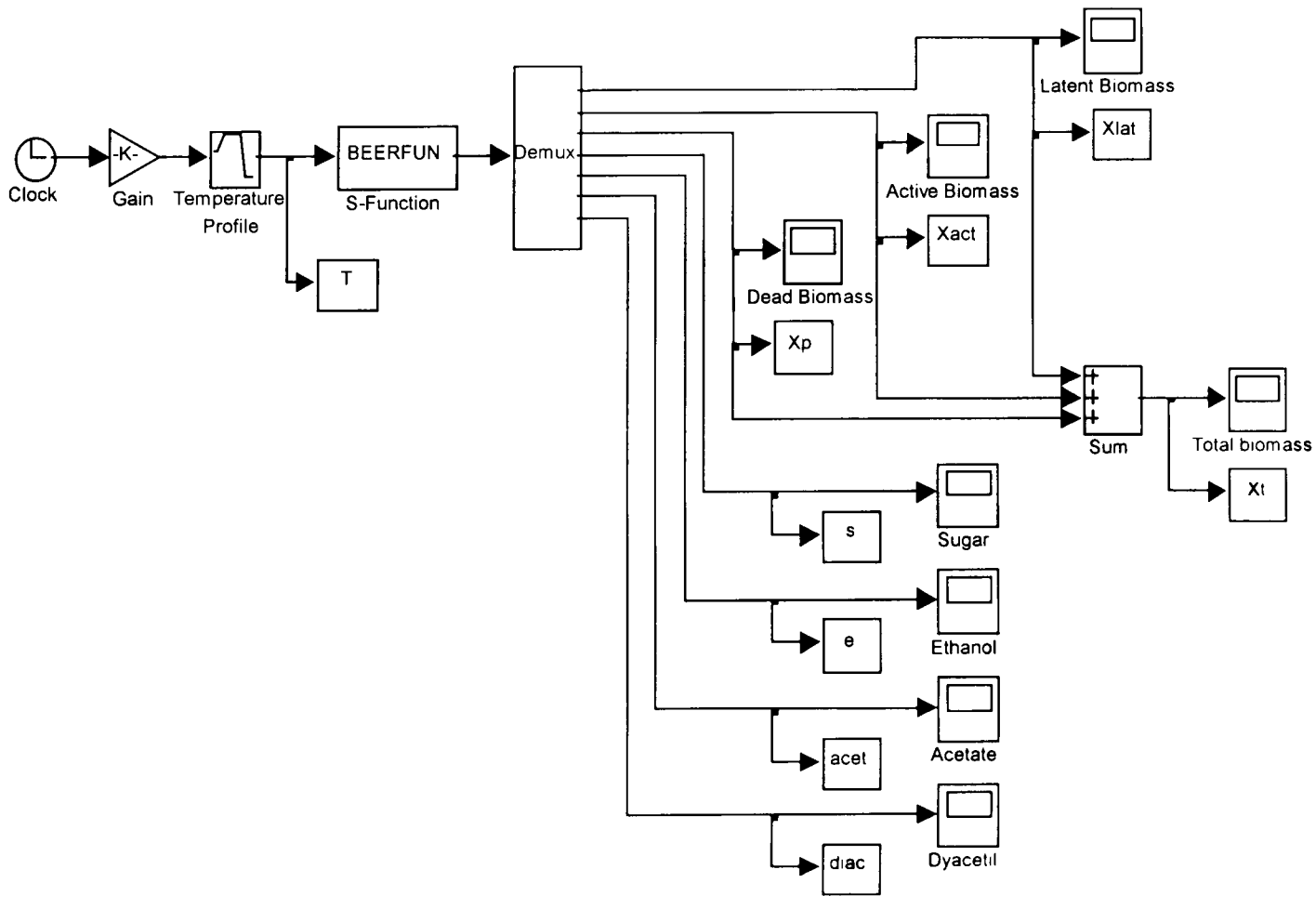

Figure 2.3 SIMULINK Model of the Beer Fermentation Process

For the initial temperature reference for the simulation, the temperature profile used in the beer company investigated by Andres-Toro et al (1998) has been selected. To represent this profile, a vector that includes the coordinate temperature values and other one with its corresponding time values have been used. Interpolation has been selected and performed between these two vectors with the assistance of the look-up table that includes the values. This temperature profile can be seen in Figure 2.4.

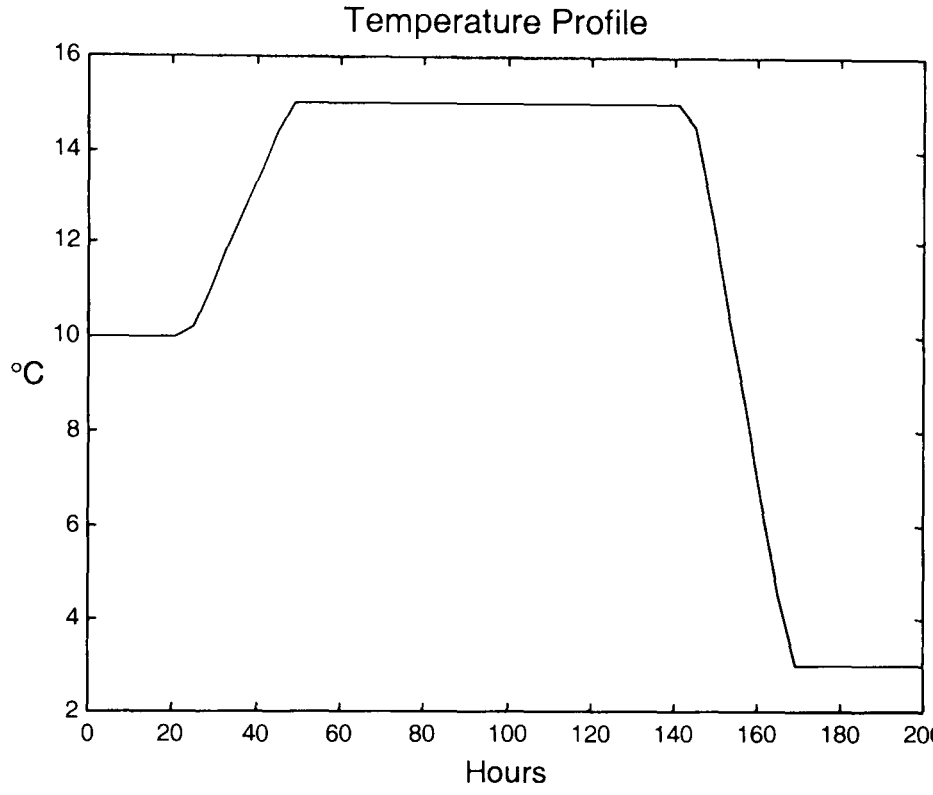

Figure 2.4 Temperature Profile used in the Beer Industry

With this temperature profile, the output variables of the process can be obtained and plotted directly from the SIMULINK model. It is important to note that the total time for the fermentation process to be modelled has been set to 160 hours, in spite of the maximum time allowed by the industry that can be seen in Figure 2.4 (200 hours). This reduction of the fermentation time has been carried out after experimentation applying different temperature profiles to the mathematical model, noticing that between 120 and 200 hours the sugar consumption and ethanol concentration tend to achieve a nearly unchanged value. Herewith, an average time of 160 hours has been selected to be the total time for the beer fermentation ($(120+200)/2=160$). With this, a reduction in the overall time while attaining the same outcome can be pursued as seen in the literature by Andres-Toro et al (1999) using a multi-objective function. For the particular case of this thesis, a minimisation

of time has not been included as part of this research, leaving the possibility of further work in this area.

The behaviour of the output variables is shown in the subsequent figures 2.5-2.9.

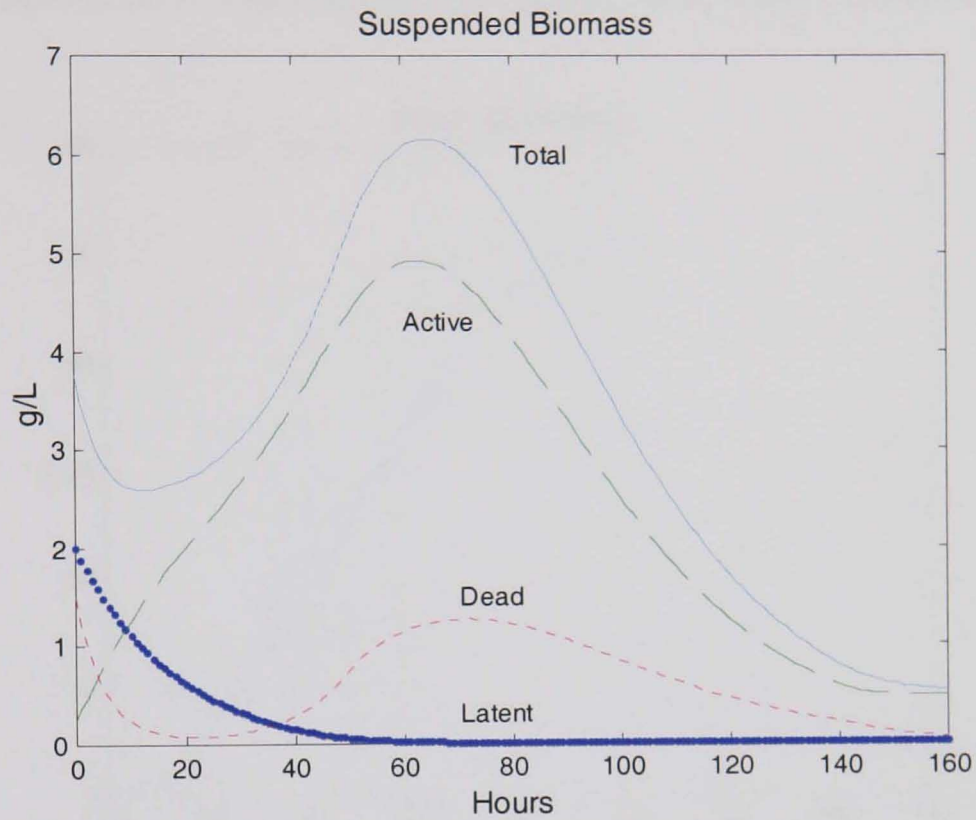

Figure 2.5 Suspended Biomass behaviour (x)

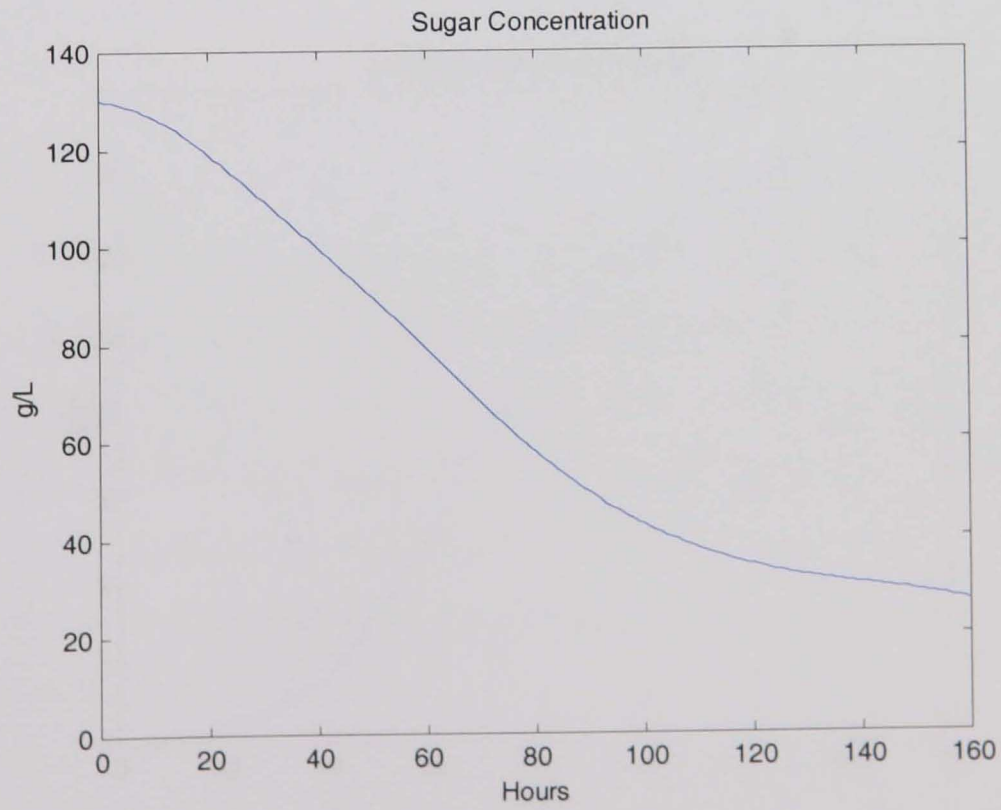

Figure 2.6 Sugar behaviour (s)

The suspended biomass behaviour in Figure 2.5 includes the active, latent, dead and also the total biomass that is the result of the preceding three values added together. It can be distinguished from this plot of the biomass concentration and also from most of the remaining variables, shown in Figures 2.6 to 2.9, the difficulty of trying to linearise the development of the model. This should be considered as an important matter in the optimisation approaches to be applied to the beer fermentation process.

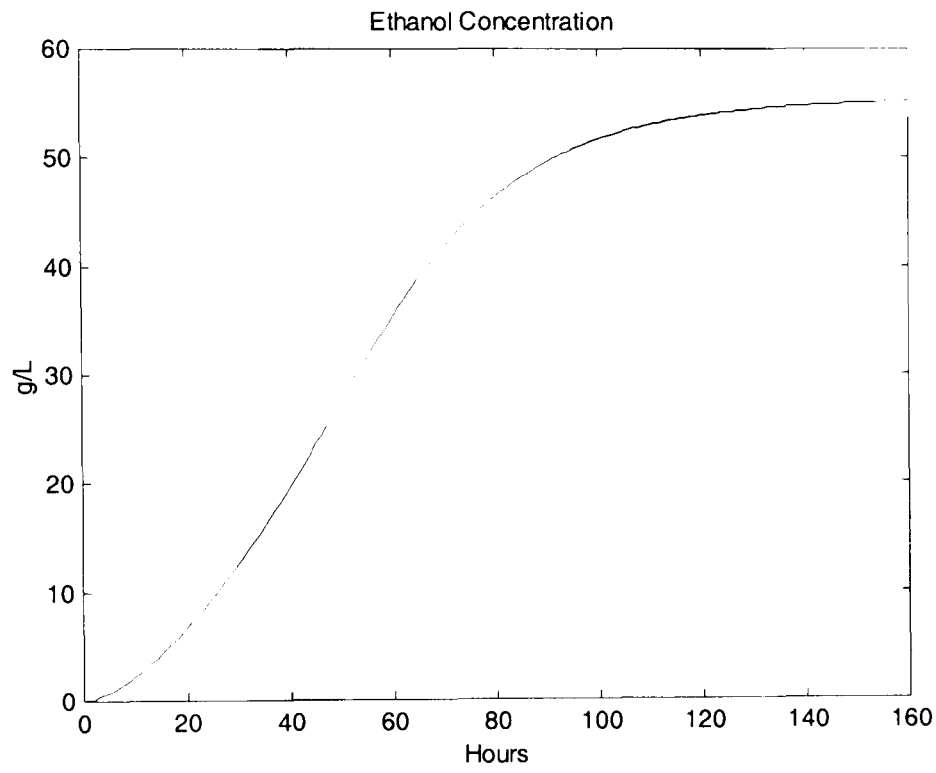

Figure 2.7 Ethanol behaviour (*e*)

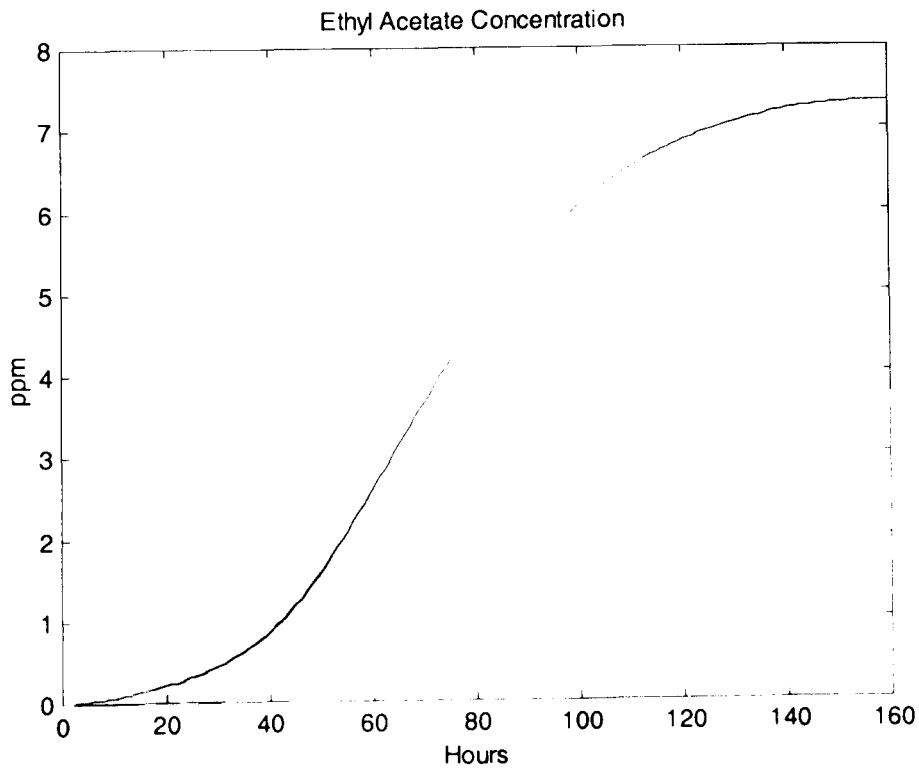

Figure 2.8 Ethyl Acetate behaviour (*acet*)

The ethanol concentration in any fermentation process is one of the most important features to be considered concerning the outcome of the process. The behaviour of this parameter is closely related to the sugar consumption and these can be seen from the two plots if placed one next to the other in order to show their evolution.

The two by-products considered in the model can be seen in Figures 2.8 and 2.9. It is important to note that only the final concentrations of these parameters affect the value of the objective function defined.

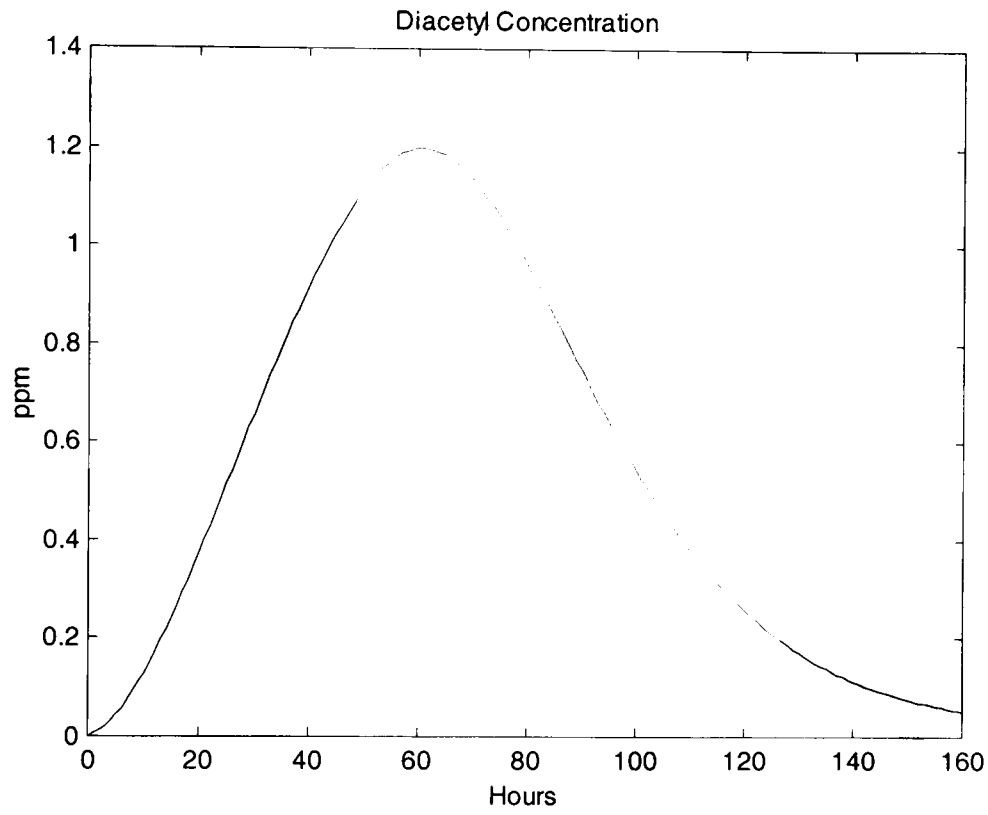

Figure 2.9 Diacetyl behaviour (*diac*)

In order to obtain the value of the objective function to be maximised by different optimisation techniques, the penalty sub functions previously defined have been included in another SIMULINK model of the process. With this an online evaluation of the objective function can be achieved. This SIMULINK model that contains the objective function value to be maximised is shown in Figure 2.10. Consequently, Table 2.5 summarises the results obtained for the different parameters related to the objective function.

As it can be seen from Table 2.5, the final values of J_2 and J_3 (which restrict diacetyl and ethyl acetate respectively) are almost negligible compared with the overall merit

of the ethanol production. This outcome have significant meaning because the exclusion of these two parameters of the objective function represent a considerable simplification of the optimisation problem.

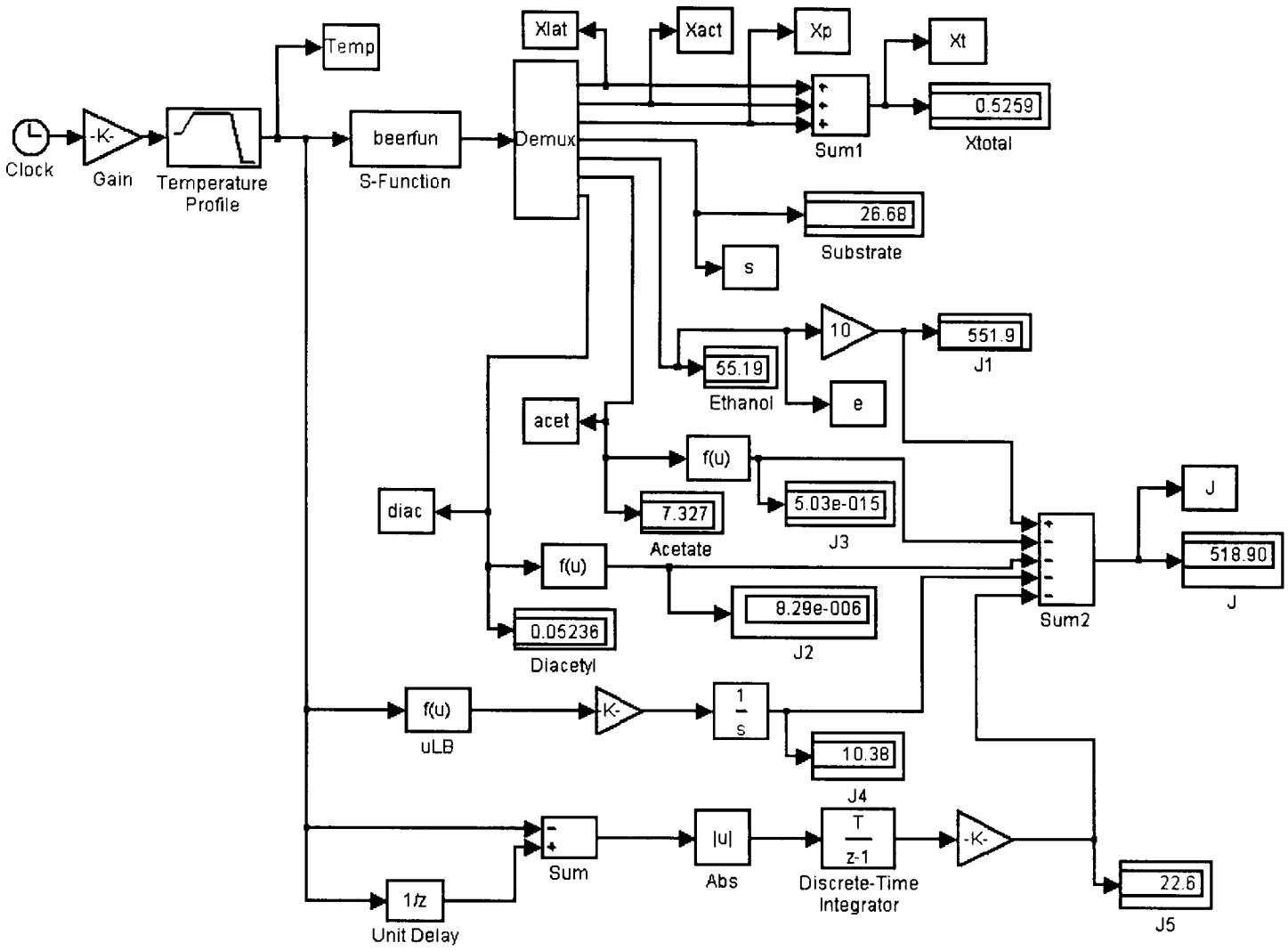

Figure 2.10 SIMULINK Model with the Objective Function

With this temperature profile, a value cost function of $J = 518.90$ has been obtained. This value will need to be optimised with the help of the optimisation techniques in the next chapters. The industry's temperature profile will serve as a starting point for some optimisation routines and the objective function value obtained with it will provide a good comparison evaluation.

Parameter	Initial value	Final value
Total Biomass (g/l)	3.75	0.5259
Sugar (g/l)	130	26.68
Ethanol (g/l)	0	55.19
Ethyl Acetate (ppm)	0	7.327

Diacetyl (ppm)	0	0.05236
J_1 (ethanol weight)	0	551.9
J_2 (restricts diacetyl)	-5.73×10^{-8}	-8.29×10^{-6}
J_3 (restricts ethyl acetate)	-1.16×10^{-29}	-5.03×10^{-15}
J_4 (avoids spoilage risk)	0	-10.38
J_5 (penalises brisk changes)	0	-22.6
J (overall value)	$\cong 0$	518.90

Table 2.5 Results after the simulation of the beer model

2.4 SUMMARY

General fermentation concepts, not just batch fermentation but fed-batch and continuous fermentation as well; have been considered in this chapter. An introduction of techniques been used to model these processes is also included. Some basic models for fermentation are included in order to show how simple or complicated fermentation processes could be. Emphasis was made on beer fermentation with batch processes; typical relationships that are part of basic modelling approaches are presented.

The use of mathematical models for real fermentation processes and different ways to implement them is included. The batch fermentation model developed by Andres-Toro et al, (1998) has been chosen among many others to become part of the simulation and optimisation research. Therefore, this model has been replicated in SIMULINK under MATLAB environment to be part of the optimisation techniques to follow in the next chapters. Simulation results have been compared with the real process results in order to be as accurate as possible and follow a real approximation of the industry's fermentation.

CHAPTER III

APPROACH TO OPTIMAL CONTROL

The initial considerations for the optimal control of the beer fermentation process presented in the previous chapter are presented below. The first section is an introduction to the DISOPE (Dynamic Integrated System Optimisation and Parameter Estimation) algorithm. Then, a concise explanation of the fermentation process itself, including the application of the Minimum Principle is made. The Gradient Method in Function Space and the DISOPE Algorithm for batch processes are introduced afterwards as valuable tools for the optimal control of the beer process selected. Then, the results obtained for different cases using these two methods are included and compared with the original values of the performance index function used by the authors of the mathematrical model.

3.1 INTRODUCTION TO THE DISOPE ALGORITHM

In practice, an exact model representation of a real process is almost unattainable; learning to cope with these discrepancies can help the control designer to achieve an optimal solution for a particular case. The ISOPE (Integrated System Optimisation and Parameter Estimation) algorithm developed by Roberts (1995) is capable of producing the true optimum regardless of model-reality differences. It takes into account the interaction between the two problems of system optimisation and parameter estimation by introducing a modifier into the model based optimisation problem.

In essence, the ISOPE technique can solve a nonlinear optimal control problem by means of iterations on a succession of properly modified reduced problems (i.e. linear quadratic problems) that include dynamic modifiers and parameters which are updated at eact iteration. One application of this kind of methods is to establish and

maintain an optimal steady-state operation of an industrial process, via the selection of regulatory controller set-point values (Roberts, 2001).

In further research, using variational calculus, the ISOPE algorithm has been extended to develop an iterative technique for solving continuous time dynamic optimal control problems, giving rise to the continuous time DISOPE algorithm (Roberts, 1993). Moreover, there are also processes in the industrial practice, which are discrete in nature and can only be controlled by using a discrete formulation of this algorithm, this has been developed, analysed and implemented by Becerra and Roberts (1996).

In the original problem formulation (Roberts, 1993), an unconstrained real optimal control problem (ROP) with specified initial and terminal conditions can be defined as follows:

$$\left\{ \begin{array}{ll} \min\limits_{u(t)} & \left\{ \varphi\left(x(t_f)\right) + \int_{t_0}^{t_f} L^*\left(x(t),u(t)\right)dt \right\} \\ \text{s.t.:} & \dot{x}(t) = f^*\left(x(t),u(t)\right) \; ; \;\; x(t_0) = x_0 \\ & x_i(t_f) = x_{if} \;\; , \;\; i \in [1,q] \;\; , \;\; q < n \end{array} \right\} \qquad (3.1)$$

where $t \in [t_0 \, , \, t_f]$ is the fixed time interval; $u(t) \in \Re^m$ and $x(t) \in \Re^n$ are the continuous control and state vectors respectively; $\varphi : \Re^{n-q} \rightarrow \Re$ is a scalar valued terminal measure; $L^* : \Re^n \times \Re^m \rightarrow \Re$ is the real performance function; and finally f^*: $\Re^n \times \Re^m \rightarrow \Re^n$ represents a set of equations describing the process.

Nevertheless, instead of solving the previous ROP of equation 3.1, the subsequent perhaps simplified model based problem (MOP) can be taken into account:

$$\left\{ \begin{array}{ll} \min\limits_{u(t)} & \left\{ \varphi\left(x(t_f)\right) + \int_{t_0}^{t_f} L\left(x(t),u(t),\gamma(t)\right)dt \right\} \\ \text{s.t.:} & \dot{x}(t) = f\left(x(t),u(t),\alpha(t)\right) \; ; \;\; x(t_0) = x_0 \\ & x_i(t_f) = x_{if} \;\; , \;\; i \in [1,q] \;\; , \;\; q < n \end{array} \right\} \qquad (3.2)$$

where $\gamma(t) \in \Re$ and $\alpha(t) \in \Re^r$ are continuous paremeters; $L : \Re^n \times \Re^m \rightarrow \Re$ is the model performance function; and correspondingly $f : \Re^n \times \Re^m \times \Re^r \rightarrow \Re^n$ represents an approximated dynamic model.

By using a two-step method, the solution of the MOP from equation 3.2 provides the control $u(t)$ as a function of the existing parameter estimates $\alpha(t) = \alpha(u(t))$ and $\gamma(t) = \gamma(u(t))$. These estimates can be attained by matching model and real states and the performance at the current calculated control $u(t) = u(\alpha(t), \gamma(t))$. Herewith, the two problems of optimisation and parameter estimation can work together to obtain (in several iterations because the model is an approximation of reality) convergence to the final solution. Even with this, in most of the cases the simple iteration between successive solutions do not lead to the right solution of the ROP.

The DISOPE technique for continuous time integrates system optimisation with parameter estimation in order to define an expanded optimal control problem (EOP). which is equivalent to an unconstrained real optimal control problem (ROP).

$$\left\{ \begin{array}{l} \min\limits_{u(t)} \left\{ \varphi\big(x(t_f)\big) + \int\limits_{t_0}^{t_f} \left[L\big(x(t),u(t),\gamma(t)\big) + \tfrac{1}{2} r^1 \left\| u(t) - v(t) \right\|^2 + \tfrac{1}{2} r^2 \left\| x(t) - z(t) \right\|^2 \right] dt \right\} \\ \\ \text{s.t.:} \quad \dot{x}(t) = f\big(x(t),u(t),\alpha(t)\big) \; ; \; x(t_0) = x_0 \\ \qquad\qquad\qquad\qquad\qquad\qquad x_i(t_f) = x_{if} \quad , \quad i \in [1,q] \\ \qquad\qquad\qquad\qquad f\big(z(t),v(t),\alpha(t)\big) = f^*\big(z(t),v(t)\big) \\ \qquad\qquad\qquad\qquad L\big(z(t),v(t),\gamma(t)\big) = L^*\big(z(t),v(t)\big) \\ \qquad\qquad\qquad\qquad\qquad\qquad u(t) = v(t) \\ \qquad\qquad\qquad\qquad\qquad\qquad x(t) = z(t) \end{array} \right\} \quad (3.3)$$

where $v(t) \in \Re^m$ and $z(t) \in \Re^n$ have been included to differentiate between the control and states in the optimal control from the relevant signals in the parameter estimation problem and the application to reality. The convexification terms $\left\| u(t) - v(t) \right\|^2$ and $\left\| x(t) - z(t) \right\|^2$ have been incorporated to convexify the performance index to facilitate convergence.

By means of Lagrange multipliers, all the equality constraints and the performance index from equation 3.3 can be added together. Consequently, an augmented Hamiltonian function can be defined as follows:

$$H(t) = L\big(x(t),u(t),\lambda(t)\big) + p^T(t) f\big(x(t),u(t),\alpha(t)\big) - \lambda(t)u(t) - \beta^T(t)x(t) \qquad (3.4)$$

where $p(t) \in \Re^n$ is a multiplier function called the costate vector; as well as $\lambda(t) \in \Re^m$ and $\beta(t) \in \Re^n$ are modifier functions. Applying calculus of variations to equation 3.3 and by means of equation 3.4 and some algebraic manipulation, the following subsets of the necessary optimality conditions can be created:

$$\left\{ \begin{array}{r} \nabla H_u(.) + r_1\big(u(t)\big) - v(t) = 0 \\ \nabla H_x(.) + \dot{p}(t) + r_2\big(x(t)\big) - z(t) = 0 \\ \nabla H_p(.) + \dot{x}(t) = 0 \end{array} \right\} \qquad (3.5)$$

With this, the boundary conditions can be defined as:

$$\left\{ \begin{array}{l} x(t_0) = x_0 \\ x_i(t_f) = x_{if} \quad , \quad i \in [1,q] \\ p_i(t_f) = \nabla\phi\big(x(t_f)\big) \quad , \quad i \in [q+1,n] \end{array} \right\} \qquad (3.6)$$

The equality constraints previously stated in the formulation of the EOP have to be considered:

$$\left\{ \begin{array}{l} f\big(z(t),v(t),\alpha(t)\big) - f^*\big(z(t),v(t)\big) = 0 \\ L\big(z(t),v(t),\gamma(t)\big) - L^*\big(z(t),v(t)\big) = 0 \end{array} \right\} \qquad (3.7)$$

$$\left\{ \begin{array}{l} v(t) = u(t) \\ z(t) = x(t) \end{array} \right\} \qquad (3.8)$$

And thus, the multiplier function equations $\lambda(t)$ and $\beta(t)$ are as follows:

$$\left\{ \begin{array}{l} \lambda(t) = \left[\dfrac{\partial f}{\partial v}(.) - \dfrac{\partial f^*}{\partial v}(.)\right]^T p(t) + \big(\nabla_v L^*(.) - \nabla_v L(.)\big) \\ \\ \beta(t) = \left[\dfrac{\partial f}{\partial z}(.) - \dfrac{\partial f^*}{\partial z}(.)\right]^T p(t) + \big(\nabla_z L^*(.) - \nabla_z L(.)\big) \end{array} \right\} \qquad (3.9)$$

The augmented Hamiltonian equation 3.4 and the optimality conditions from equation 3.5 can be fulfilled by solving a modified model based optimal control problem (MMOP) that can be described in this way:

$$\left\{\begin{array}{l} \min\limits_{u(t)} \left\{\varphi\left(x(t_f)\right)+\int_{t_0}^{t_f}\left[\begin{array}{l} L\left(x(t),u(t),\gamma(t)\right)-\lambda(t)u(t)-\beta^T(t)x(t) \\ +\tfrac{1}{2}r^1\left\|u(t)-v(t)\right\|^2+\tfrac{1}{2}r^2\left\|x(t)-z(t)\right\|^2 \end{array}\right]dt\right\} \\ \text{s.t.:} \quad \dot{x}(t)=f\left(x(t),u(t),\alpha(t)\right)\ ;\ x(t_0)=x_0 \\ \qquad x_i(t_f)=x_{if}\ ,\ i\in[1,q] \end{array}\right. \tag{3.10}$$

where the parameters in this equation have previously been specified. In relation to the optimality conditions in equation 3.7, it can be said that they describe the parameter estimation problem and the ones in equation 3.8 the association constraints between the parameter estimation problem and the MMOP. Meanwhile, equation 3.9 defines the calculations required for computing the modifiers.

The previous reasoning leads to the consequent description of the DISOPE algorithm whose iterations, assuming convergence, reach the solution of the ROP by means of recurring solutions of the MMOP (Roberts, 1993):

Data: $f(.)$, $L(.)$, x_0, x_{if} ($i \in [1,q]$), $\varphi(.)$, t_0, t_f, r_1, r_2, and means for calculating $f^*(.)$, $L^*(.)$.

Step 0: *initialisation*: Compute or assume a nominal solution $u(t)^0$, $x(t)^0$ and $p(t)^0$.

Step 1: *application to reality*: Set iteration counter $i = 0$, $v(t)^0 = u(t)^0$, $z(t)^0 = x(t)^0$; $t \in [t_0, t_f]$.

Step 2: *parameter estimation*: Compute $\alpha(t)^i$ and $\gamma(t)^i$ to satisfy equation 3.7.

Step 3: *modifiers*: Compute $\lambda(t)^i$ and $\beta(t)^i$ from equation 3.9. This requires finding the derivatives with respect to v and z, of $f(.)$, $f^*(.)$, $L(.)$ and $L^*(.)$.

Step 4: *system optimisation*: With specified $\alpha(t)^i$, $\gamma(t)^i$, $\lambda(t)^i$, $v(t)^i$, $\beta(t)^i$ and $z(t)^i$: solve the MMOP defined in equation 3.10 to obtain $\hat{u}(t)^i$, $\hat{x}(t)^i$ and $\hat{p}(t)^i$.

Step 5: *update*: Test convergence and update the estimate for the optimal solution of the ROP. Additionally to the convexification terms from equation 3.3

and regulated by choice of r_1 and r_2; a simple relaxation method is used to satisfy equation 3.8. With this:

$$\begin{aligned} v(t)^{i+1} &= v(t)^i + k_v\left(\hat{u}(t)^i - v(t)^i\right) \\ z(t)^{i+1} &= z(t)^i + k_z\left(\hat{x}(t)^i - z(t)^i\right) \\ p(t)^{i+1} &= p(t)^i + k_p\left(\hat{p}(t)^i - p(t)^i\right) \end{aligned} \tag{3.11}$$

where k_v, k_z, $k_p \in [0,1]$ are scalar gains. If $v(t)^{i+1} = v(t)^i$, within a defined tolerance then stop, else set $i = i+1$ and continue from **Step 2**.

For some problems in practice, reaching the equality $v(t)^{i+1} = v(t)^i$ with $t \in [t_0, t_f]$, can be evaluated by using the following 2-norm (control variation norm between iterates):

$$\left\| v^{i+1} - v^i \right\|_2 = \sqrt{\frac{1}{\Delta t}\int_{t_0}^{t_f} \left\| v(t)^{i+1} - v(t)^i \right\|^2 dt} \tag{3.12}$$

where Δt is the numerical integration step and this value can be compared with a given small tolerance ε_v defined a priori.

It is important to note that the required derivatives from equation 3.9 may be difficult to measure in practice. For this reason, it might be necessary to opt for perturbation and finite difference calculations in order to obtain these values at every titeration. A different way is to approximate them iteration by iteration using Broyden's method as described by Roberts (2000).

The DISOPE Algorithm has effectively been applied to fed-batch control processes (Becerra and Roberts, 1998a); incorporated in the receding horizon optimal control strategy of model predictive control (Becerra and Roberts, 1998b); and most recently the ISOPE approach has been used to design a DISOPE technique for optimal control of differential and algebraic equation (DAE) systems (Roberts and Becerra, 2000).

3.2 OPTIMISATION OF THE BEER FERMENTATION PROCESS

The optimisation of the beer fermentation process selected in the previous chapter is now pursued. With this in mind, the mathematical model of the fermentation process is included with a description of each of the parameters used in Table 3.1.

Parameter	Description
μ_a	Ethanol production rate
μ_D	Specific yeast settling down rate
μ_{eas}	Ethyl acetate coefficient rate
μ_{lag}	Specific rate of latent formation
μ_s	Substrate consumption rate
μ_x	Specific yeast growth rate
$acet$	Ethyl acetate concentration
$diac$	Diacetyl concentration
e	Ethanol concentration
f	Fermentation inhibitor factor
k_{dc}	Diacetyl appearance rate
k_{dm}	Diacetyl reduction rate
k_m	Yeast growth inhibition parameter
k_s	Sugar inhibition parameter
s	Concentration of sugar
s_0	Initial concentration of sugar
t	Time
T	Temperature
x_{active}	Suspended active biomass
x_{dead}	Suspended dead biomass
x_{lag}	Suspended latent biomass

Table 3.1 Nomenclature used

The beer fermentation process can be defined by the subsequent mathematical equations:

$$\mu_x = \frac{\mu_{x0} \cdot s}{0.5 \cdot s_0 + e} \tag{3.13}$$

$$\mu_D = \frac{0.5 \cdot s_0 \cdot \mu_{D0}}{0.5 \cdot s_0 + e} \tag{3.14}$$

$$\mu_s = \frac{\mu_{s0} \cdot s}{k_s + s} \tag{3.15}$$

$$\mu_a = \frac{\mu_{a0} \cdot s}{k_s + s} \tag{3.16}$$

$$f = 1 - \frac{e}{0.5 \cdot s_0} \tag{3.17}$$

$$\frac{dx_{lag}}{dt} = -\mu_{lag} \cdot x_{lag} \tag{3.18}$$

$$\frac{dx_{active}}{dt} = \mu_x \cdot x_{active} - k_m \cdot x_{active} + \mu_{lag} \cdot x_{lag} \tag{3.19}$$

$$\frac{dx_{dead}}{dt} = k_m \cdot x_{active} - \mu_D \cdot x_{dead} \tag{3.20}$$

$$\frac{ds}{dt} = -\mu_s \cdot x_{active} \tag{3.21}$$

$$\frac{de}{dt} = \mu_a \cdot f \cdot x_{active} \tag{3.22}$$

$$\frac{d(acet)}{dt} = \mu_{eas} \cdot \mu_s \cdot x_{active} \tag{3.23}$$

$$\frac{d(diac)}{dt} = k_{dc} \cdot s \cdot x_{active} - k_{dm} \cdot diac \cdot e \tag{3.24}$$

where

$$\mu_{x0} = 1.095 \times 10^{47} \cdot e^{\frac{-63720}{R \cdot (T + T_a)}} \qquad k_m = 3.373 \times 10^{56} \cdot e^{\frac{-76450}{R \cdot (T + T_a)}}$$

$$\mu_{eas} = 1.129 \times 10^{39} \cdot e^{\frac{-53056}{R \cdot (T + T_a)}} \qquad \mu_{D0} = 4.889 \times 10^{14} \cdot e^{\frac{-20020}{R \cdot (T + T_a)}}$$

$$\mu_{s0} = 6.232 \times 10^{-19} \cdot e^{\frac{23254}{R \cdot (T + T_a)}} \qquad \mu_{a0} = 26.3865 \cdot e^{\frac{-2528.6}{R \cdot (T + T_a)}}$$

$$\mu_{lag} = 2.2041 \times 10^{13} \cdot e^{\frac{-18959}{R \cdot (T + T_a)}} \qquad k_s = 1.1081 \times 10^{-52} \cdot e^{\frac{68249.2}{R \cdot (T + T_a)}}$$

$$k_{dc} = 0.000127672 \qquad k_{dm} = 0.0011386\text{4}$$

$$R = 1.99536 \qquad T_a = 273.15$$

Hence, the state space model can be defined:

$$\frac{dx_{lag}}{dt} = -\mu_{lag}(T)x_{lag} \tag{3.25}$$

$$\frac{dx_{active}}{dt} = \frac{\mu_{x0}(T)s}{0.5s_0 + e}x_{active} - k_m(T)x_{active} + \mu_{lag}(T)x_{lag} \tag{3.26}$$

$$\frac{dx_{dead}}{dt} = k_m(T)x_{active} - \frac{0.5s_0\mu_{D0}(T)}{0.5s_0 + e}x_{dead} \tag{3.27}$$

$$\frac{ds}{dt} = -\frac{\mu_{s0}(T)s}{k_s(T) + s}x_{active} \tag{3.28}$$

$$\frac{de}{dt} = \frac{\mu_{a0}(T)s}{k_s(T) + s}\left(1 - \frac{e}{0.5s_0}\right)x_{active} \tag{3.29}$$

$$\frac{d(acet)}{dt} = \mu_{eas}(T)\frac{\mu_{s0}(T)s}{k_s(T) + s}x_{active} \tag{3.30}$$

$$\frac{d(diac)}{dt} = k_{dc}sx_{active} - k_{dm}(diac)e \tag{3.31}$$

Once again, the performance index can be defined as follows:

maximise

$$J = J_1 + J_2 + J_3 + J_4 + J_5 \tag{3.32}$$

where

$$J_1 = 10 \cdot e(t_f) \tag{3.33}$$

$$J_2 = -5.73 \times 10^{-8} \cdot e^{(95 \cdot diac(t_f))} \tag{3.34}$$

$$J_3 = -1.16 \times 10^{-29} \cdot e^{(4.6 \cdot acet(t_f))} \tag{3.35}$$

$$J_4 = -\int_0^{t_f} 9.91 \times 10^{-17} \cdot e^{(2.317)} dt \tag{3.36}$$

$$J_5 = -\sum_{i=1}^{N} \frac{|T_{i+1} - T_i|}{\Delta t} \tag{3.37}$$

with $N\Delta t = t_f$

Following the simulation of the industrial profile in the previous chapter, and after several experimentation with the SIMULINK model of the process; the terms J_2 and J_3 have been found to be negligible compared with J_1. The last term J_5 has been

ignored at this stage in order to simplify the optimal control formulation and as a result, there is a significant reduction to the objective function and number of state equations. These conditions can be considered a special case for the optimisation where $J_2 = J_3 = J_5 = 0$.

A new performance index J_r is described as follows:

minimise

$$J_r = -10e(t_f) + \int_0^{t_f} 9.91\times10^{-17} \cdot e^{(2.31T)} dt \tag{3.38}$$

subject to

$$\frac{dx_{lag}}{dt} = -\mu_{lag}(T)x_{lag} \tag{3.39}$$

$$\frac{dx_{active}}{dt} = \frac{\mu_{x0}(T)s}{0.5s_0 + e}x_{active} - k_m(T)x_{active} + \mu_{lag}(T)x_{lag} \tag{3.40}$$

$$\frac{ds}{dt} = -\frac{\mu_{s0}(T)s}{k_s(T) + s}x_{active} \tag{3.41}$$

$$\frac{de}{dt} = \frac{\mu_{a0}(T)s}{k_s(T) + s}\left(1 - \frac{e}{0.5s_0}\right)x_{active} \tag{3.42}$$

With this, the reduced state and control signals (x_i and u) are defined:

$$x_1 = x_{lag},\ x_2 = x_{active},\ x_3 = s,\ x_4 = e,\ u = T$$

Subsequently, replacing these variables in equations 3.38-3.42 for the special case:

$$J_r = -10x_4(t_f) + \int_0^{t_f} ce^{du} dt \tag{3.43}$$

$$\frac{dx_1}{dt} = -\mu_{lag}(u)x_1,\ \ x_1(0) = 2 \tag{3.44}$$

$$\frac{dx_2}{dt} = \frac{\mu_{x0}(u)x_3}{0.5s_0 + x_4}x_2 - k_m(u)x_2 + \mu_{lag}(u)x_1,\ \ x_2(0) = 0.5 \tag{3.45}$$

$$\frac{dx_3}{dt} = -\frac{\mu_{s0}(u)x_3}{k_s(u)+x_3}x_2, \quad x_3(0)=130 \tag{3.46}$$

$$\frac{dx_4}{dt} = \frac{\mu_{a0}(u)x_3}{k_s(u)+x_3}\left(1-\frac{x_4}{0.5s_0}\right)x_2, \quad x_4(0)=0 \tag{3.47}$$

where the given initial conditions are stated and

$$\mu_{lag}(u) = 2.2041\times10^{13}\, e^{\frac{-18959}{R(u+T_a)}} \qquad \mu_{x0}(u) = 1.095\times10^{47}\, e^{\frac{-63720}{R(u+T_a)}}$$

$$k_m(u) = 3.373\times10^{56}\, e^{\frac{-76450}{R(u+T_a)}} \qquad \mu_{s0}(u) = 6.232\times10^{-19}\, e^{\frac{23254}{R(u+T_a)}}$$

$$k_s(u) = 1.1081\times10^{-52}\, e^{\frac{68249.2}{R(u+T_a)}} \qquad \mu_{a0}(u) = 26.3865\, e^{\frac{-2528.6}{R(u+T_a)}}$$

$$c = 9.91\times10^{-17} \qquad d = 2.31$$

The optimisation problem can at this time be written in the form:

minimise

$$J_r = \Phi\left(x(t_f)\right) + \int_0^{t_f} ce^{du}dt \tag{3.48}$$

subject to

$$\dot{x} = f(x,u),\ x(0) = x_o \tag{3.49}$$

where

$$f(x,u) = \begin{bmatrix} -\mu_{lag}(u)x_1 \\ \left(\frac{\mu_{x0}(u)x_3}{0.5s_0+x_4} - k_m(u)\right)x_2 + \mu_{lag}(u)x_1 \\ -\frac{\mu_{s0}(u)x_3}{k_s(u)+x_3}x_2 \\ \frac{\mu_{a0}(u)x_3}{k_s(u)+x_3}\left(1-\frac{x_4}{0.5s_0}\right)x_2 \end{bmatrix}, \quad x(0) = \begin{bmatrix} 2 \\ 0.5 \\ 130 \\ 0 \end{bmatrix}, \quad \Phi\left(x(t_f)\right) = -10x_4(t_f)$$

3.2.1. Optimal Steady-State Solution

For this optimisation problem, an optimal steady-state solution can be attained when the input u is held constant. In this case, the optimal stedy-state solution problem is:

minimise

$$J_{ss} = \Phi\left(x(t_f)\right) + ce^{du}t_f \tag{3.50}$$

subject to

$$\dot{x} = f(x,u),\ x(0) = x_o \tag{3.51}$$

This optimal steady-state problem can readily be solved using a non-linear programming technique. For this purpose, the MATLAB routine ***fminbnd***, based on golden section search and parabolic interpolation, has been employed (Coleman et al, 1999). The model differential equation were solved using ***ode23***, which is a variable step explicit Runge-Kutta method of Bogacki and Shampine (1989). The result is:

u_{ss} = 14.6211°C with a performance index of J_{ss} = -597.8082

The resultant state response signals are shown in Figures 3.1 to 3.4.

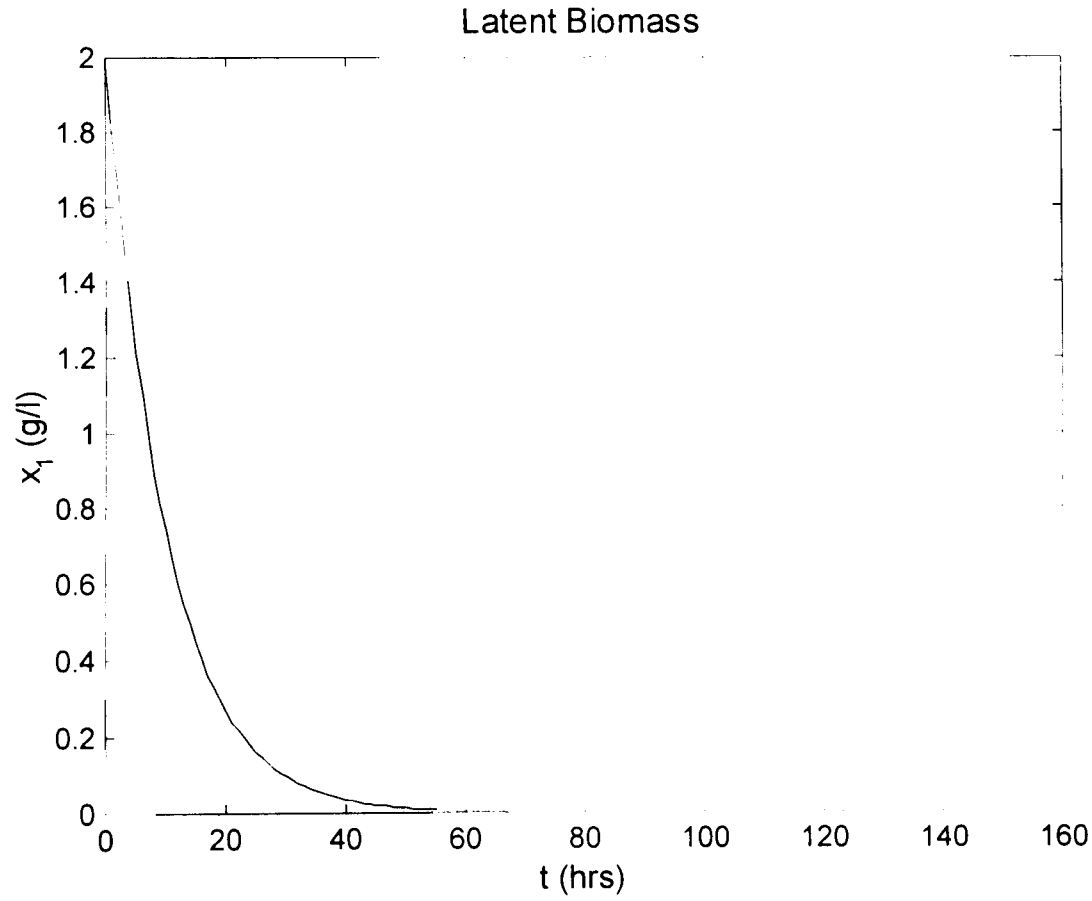

Figure 3.1 State response x_1 for the optimum steady-state profile

Accordingly, this result of u_{ss}=14.6211°C has then been applied as the temperature input profile of the full SIMULINK model of the fermentation process. This has been done in order to verify the performance values obtained for each one of the

parameters of the objective function (J_1, J_2,..., J_5); the outcome is included in Fig. 3.5 and summarised in Table 3.2.

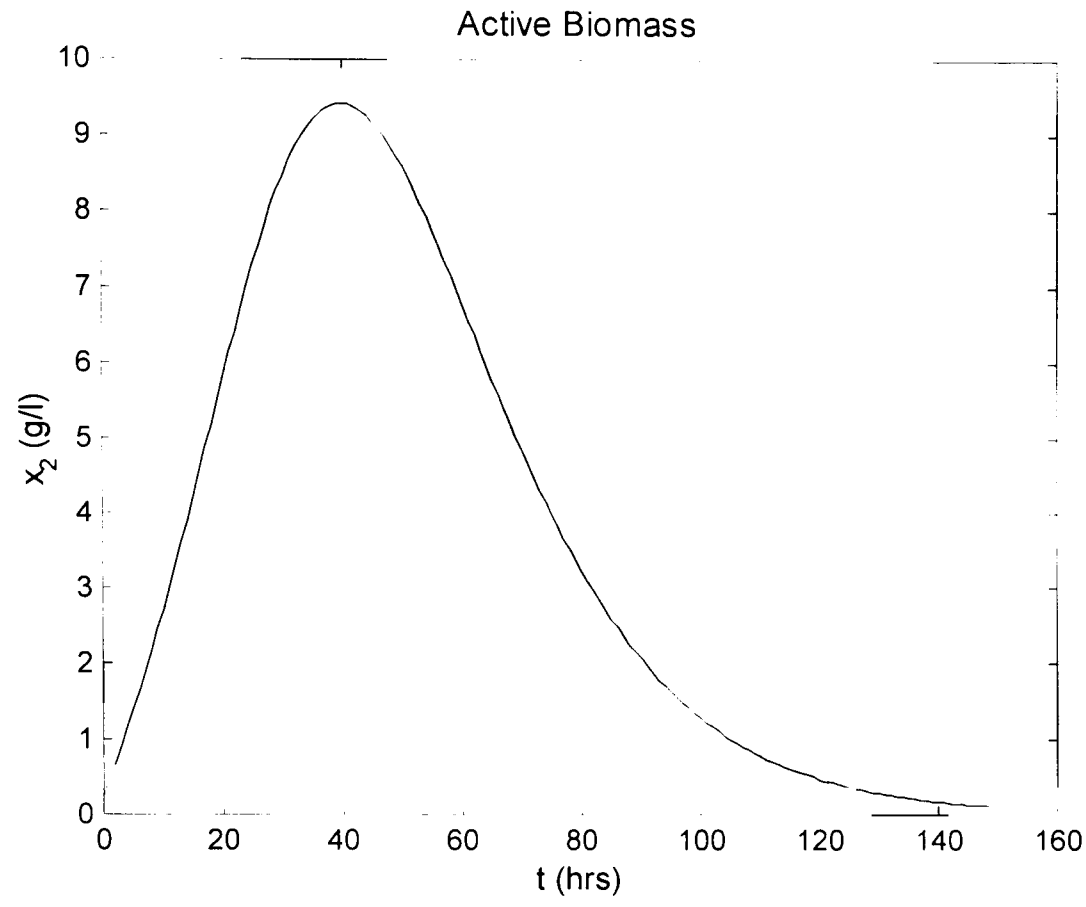

Figure 3.2 State response x_2 for the optimum steady-state profile

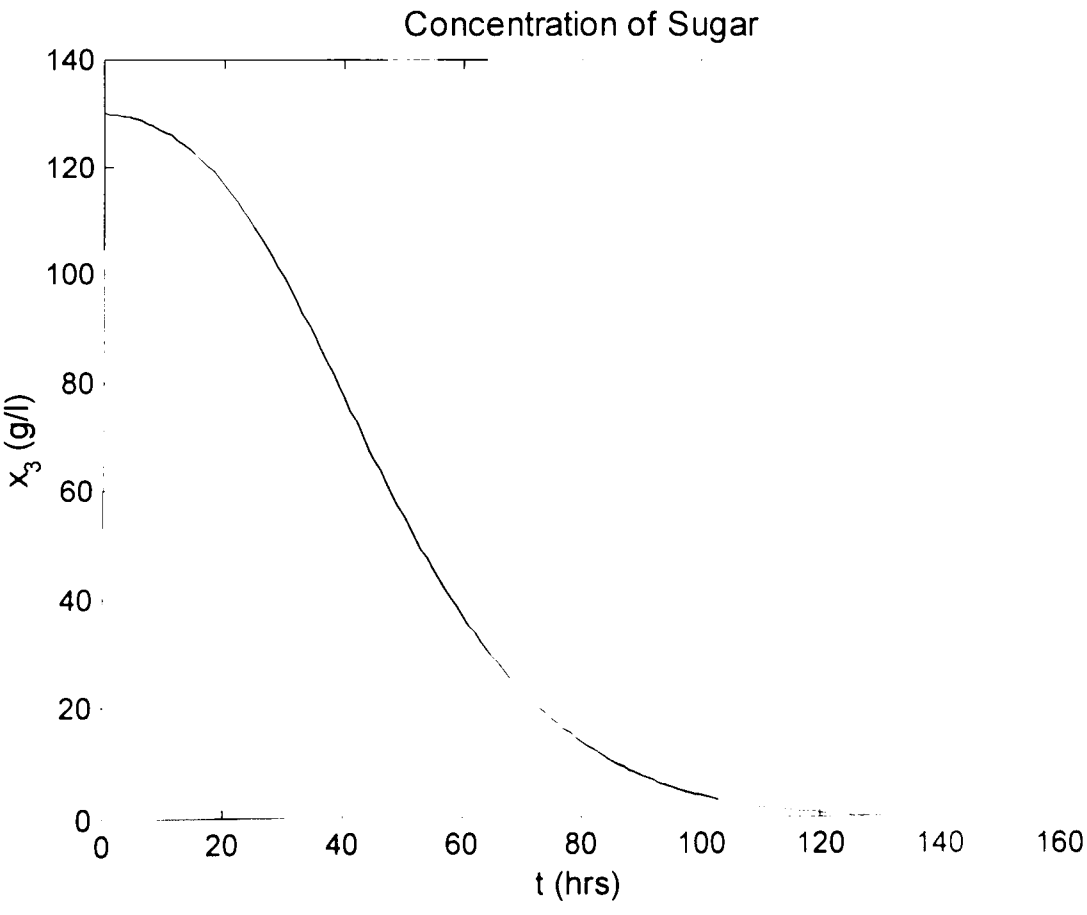

Figure 3.3 State response x_3 for the optimum steady-state profile

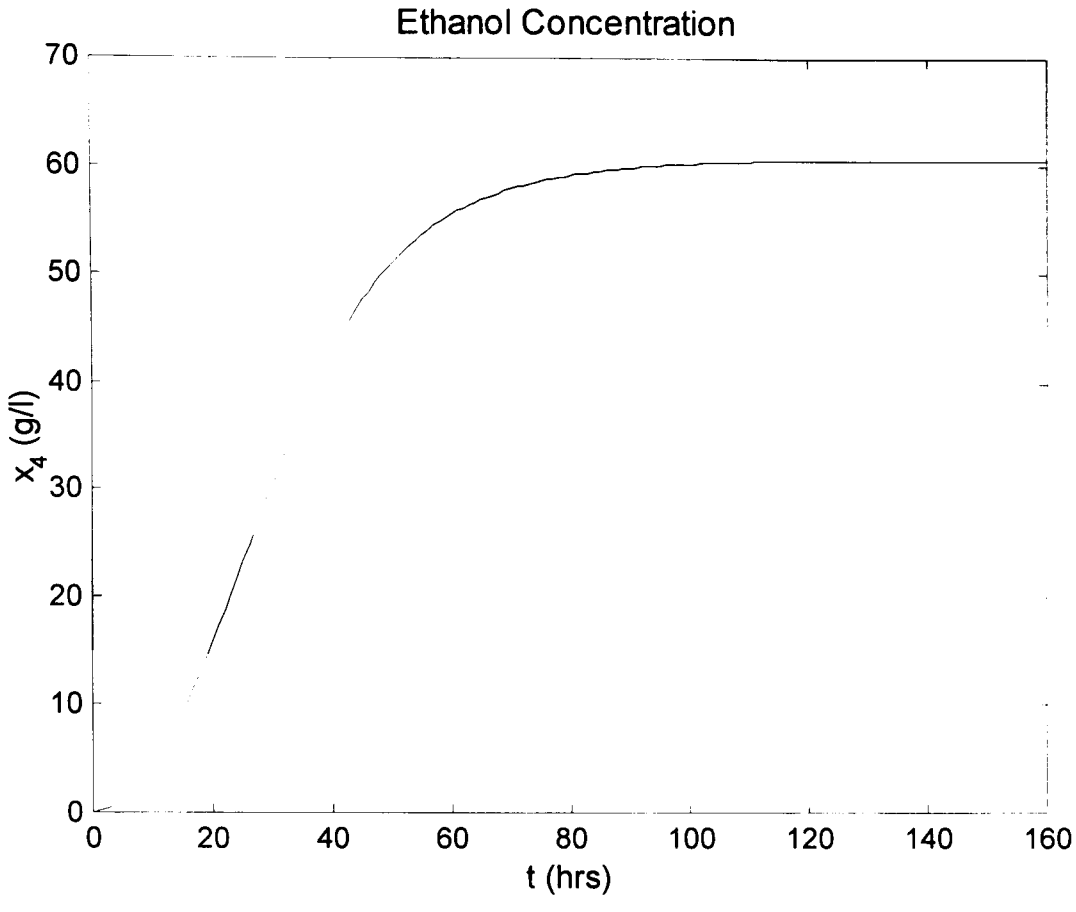

Figure 3.4 State response x_4 for the optimum steady-state profile

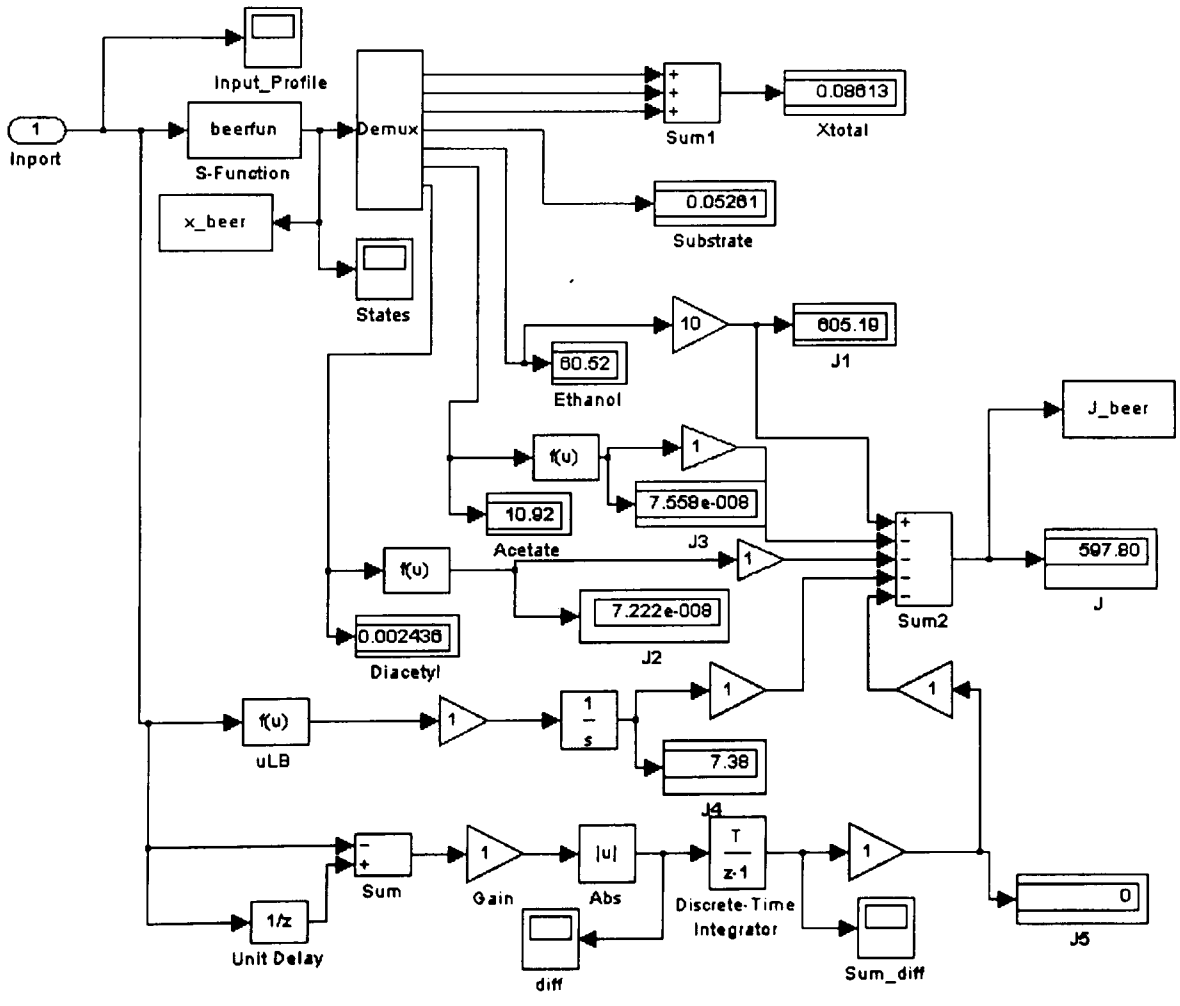

Figure 3.5 SIMULINK Model for the steady-state case

The SIMULINK model for the optimum steady-state profile including the final values of the equation variables and performance sub functions can be seen in Figure 3.5.

Parameter	Final value
Total Biomass (g/l)	0.0861
Sugar (g/l)	0.0526
Ethanol (g/l)	60.52
Ethyl Acetate (ppm)	10.92
Diacetyl (ppm)	0.0024
J_1 (ethanol weight)	605.19
J_2 (restricts diacetyl)	-7.22×10^{-8}
J_3 (restricts ethyl acetate)	-7.56×10^{-8}
J_4 (avoids spoilage risk)	-7.38
J_5 (penalises brisk changes)	0
J (overall value)	597.80

Table 3.2 Results corresponding to optimum steady-state control profile

As it can be observed from this result with the optimum steady-state control profile, an increment in the performance index from the original 518.90 with the industry's temperature profile has been reached to a new 597.80 value. This means a percentage increase of more than 15% by making the problem formulation simpler.

3.2.2. Application of Calculus of Variations

The standard necessary optimality conditions for an unconstrained optimal control problem can be found by means of calculus of variations. This principle is now applied to the optimisation problem (Lewis and Syrmos, 1995) and thus, the Hamiltonian (H) can be defined as:

$$H = cc^{du} + p^T f(x,u) \qquad (3.52)$$

Therefore, the Hamiltonian Gradient can be represented as follows:

$$H_u = \frac{dH}{du} = cde^{du} + \left[\frac{\partial f}{\partial u}\right]^T p \tag{3.53}$$

Which equals to zero to obtain a minimum when:

$$u = \frac{\log_e\left(-\dfrac{\left[\dfrac{\partial f}{\partial u}\right]^T p}{cd}\right)}{d} \tag{3.54}$$

It is important to note that since c and d are both positive, the equation 3.54 will only give a valid (real) result for u when:

$$\left[\frac{\partial f}{\partial u}\right]^T p < 0 \tag{3.55}$$

The co-state equations are:

$$\dot{p} = -\nabla_x H,\ p(t_f) = \left.\frac{\partial \Phi}{\partial x}\right|_{x=x(t_f)} \tag{3.56}$$

which gives as a result:

$$\dot{p} = -\left[\frac{\partial f}{\partial x}\right]^T p,\ p(t_f) = \begin{bmatrix} 0 & 0 & 0 & -10 \end{bmatrix}^T \tag{3.57}$$

In equations 3.57 and 3.53, $\left[\dfrac{\partial f}{\partial x}\right]$ and $\left[\dfrac{\partial f}{\partial u}\right]$ are the Jacobian (A and B) matrices given by

$$A(x,u)=\left[\frac{\partial f}{\partial x}\right]=\begin{bmatrix} a_{1,1}(u) & 0 & 0 & 0 \\ a_{2,1}(u) & a_{2,2}(x,u) & a_{2,3}(x,u) & a_{2,4}(x,u) \\ 0 & a_{3,2}(x,u) & a_{3,3}(x,u) & 0 \\ 0 & a_{4,2}(x,u) & a_{4,3}(x,u) & a_{4,4}(x,u) \end{bmatrix} \tag{3.58}$$

$$B(x,u)=\left[\frac{\partial f}{\partial u}\right]=\begin{bmatrix} b_1(x,u) \\ b_2(x,u) \\ b_3(x,u) \\ b_4(x,u) \end{bmatrix} \tag{3.59}$$

Then, the value of the different terms of these matrices can be defined:

$$a_{1,1}(u)=-\mu_{lag}(u) \qquad a_{2,1}(u)=\mu_{lag}(u)$$

$$a_{2,2}(x,u)=\frac{\mu_{x0}(u)x_3}{0.5s_0+x_4}-k_m(u) \qquad a_{2,3}(x,u)=\frac{\mu_{x0}(u)x_2}{0.5s_0+x_4}$$

$$a_{2,4}(x,u)=-\frac{\mu_{x0}(u)x_2x_3}{\left(0.5s_0+x_4\right)^2} \qquad a_{3,2}(x,u)=-\frac{\mu_{s0}(u)x_3}{k_s(u)+x_3}$$

$$a_{3,3}(x,u)=-\frac{\mu_{s0}(u)k_s(u)x_2}{\left(k_s(u)+x_3\right)^2} \qquad a_{4,2}(x,u)=\frac{\mu_{a0}(u)\left(1-\frac{2x_4}{s_0}\right)x_3}{k_s(u)+x_3}$$

$$a_{4,3}(x,u)=\frac{\mu_{a0}(u)k_s(u)\left(1-\frac{2x_4}{s_0}\right)x_2}{\left(k_s(u)+x_3\right)^2} \qquad a_{4,4}(x,u)=-\frac{2\mu_{a0}(u)x_2x_3}{s_0\left(k_s(u)+x_3\right)}$$

$$b_1(x,u)=-\mu_{lag}^{'}(u)x_1$$

$$b_2(x,u)=\mu_{lag}^{'}(u)x_1+\left(\frac{\mu_{x0}^{'}(u)x_3}{0.5s_0+x_4}-k_m^{'}(u)\right)x_2$$

$$b_3(x,u)=-x_2x_3\,\frac{\left(\mu_{s0}^{'}(u)\left(k_s(u)+x_3\right)-\mu_{s0}(u)k_s^{'}(u)\right)}{\left(k_s(u)+x_3\right)^2}$$

$$b_4(x,u)=x_2x_3\left(1-\frac{2x_4}{s_0}\right)\left(\frac{\mu_{a0}^{'}(u)\left(k_s(u)+x_3\right)-\mu_{a0}(u)k_s^{'}(u)}{\left(k_s(u)+x_3\right)^2}\right)$$

where

$$\mu_{lag}^{'}(u)=\frac{\partial \mu_{lag}(u)}{\partial u}=\frac{2.2041\times10^{13}\times18959e^{\frac{-18959}{R(u+T_a)}}}{R(u+T_a)^2}$$

$$\mu_{x0}^{'}(u)=\frac{\partial \mu_{x0}(u)}{\partial u}=\frac{1.095\times10^{47}\times63720e^{\frac{-63720}{R(u+T_a)}}}{R(u+T_a)^2}$$

$$k_{m}^{'}(u)=\frac{\partial k_{m}(u)}{\partial u}=\frac{3.373\times10^{56}\times76450e^{\frac{-76450}{R(u+T_a)}}}{R(u+T_a)^2}$$

$$\mu_{s0}^{'}(u)=\frac{\partial \mu_{s0}(u)}{\partial u}=-\frac{6.232\times10^{-19}\times23254e^{\frac{23254}{R(u+T_a)}}}{R(u+T_a)^2}$$

$$k_{s}^{'}(u)=\frac{\partial k_{s}(u)}{\partial u}=-\frac{1.1081\times10^{-52}\times68249.2e^{\frac{68249.2}{R(u+T_a)}}}{R(u+T_a)^2}$$

$$\mu_{a0}^{'}(u)=\frac{\partial \mu_{a0}(u)}{\partial u}=\frac{26.3865\times2528.6e^{\frac{-2528.6}{R(u+T_a)}}}{R(u+T_a)^2}$$

By replacing these expressions in its corresponding places, the matrices A and B are defined.

3.3 GRADIENT METHOD IN FUNCTION SPACE

In order to find the numerical computation of optimal control of both continuous and discrete-time systems there are mainly two approaches, according to Noton (1972): indirect and direct methods of minimisation of a performance index (objective function). In the direct approaches to minimisation, the state equations are influenced only by the control *u(t)* and the minimisation of the objective function *J(u)* by direct adjustment of *u(t)* is sought. For the particular case of the continuous process that the beer fermentation presents, a direct method known as the Gradient Method in Function Space has been selected.

A method of first order gradients is the simplest approach to a direct method in which the state and co-state equations remain separated. The logical and convenient approach is to regard continuous time systems as the limiting case of discrete systems as the subinterval of time tends to zero.

This method is probably the easiest and most stable computation of optimal control, because if the system state equations are stable in forward time then the co-state equations are stable when integrated in reverse time. Convergence is not usually critically dependent on a first approximation.

The method of steepest descent (first order gradients) consists therefore of applying the iterative correction:

$$\Delta u(t) = -e\frac{\partial H}{\partial u(t)} = -eH_u \qquad (3.60)$$

where e is an arbitrary constant provided to regulate convergence (stability). Since H depends on the co-state variables, the $f(t)$ have to be generated by an iterative process.

Thus, the algorithm can be described as follows:

Data: t_f, e

***Step* 0**: *initialisation*: Set iteration counter $k = 1$, and guess $u(t)^0$ for the interval $[0,t_f]$.

***Step* 1**: Compute $x(t)$ and J_r from equations 3.49 and 3.48.

***Step* 2**: Compute the co-state vector $p(t)$ by solving equation 3.57 in reverse time.

***Step* 3**: Compute the Hamiltonian gradient H_u from equation 3.53.

***Step* 4**: Update $u(t)$ using equation 3.60. That is:

$$u(t)^{k+1} = u(t)^k - eH_u \tag{3.61}$$

Set $k = k+1$ and repeat from ***Step 1*** until convergence is achieved.

Note that because equation 3.54 is not solved explicity, the associated validity condition is not an issue.

The following three cases have been taken into consideration:

Case 1: *u(t)* has been initialised at the optimum steady-state solution, that is:

$$u(t) = 14.6211,\ t \in [0,160]$$

Case 2: *u(t)* has been initialised at an arbitrary steady-state solution, that is:

$$u(t) = 10,\ t \in [0,160]$$

Case 3: *u(t)* has been initialised at the industrial profile.

For each of these cases $e = 1$ was set initially. Adaptation of this parameter was required during the optimisation routine by means of two regulating parameters k_1 and k_2 (set a priori to 1.1 and 0.75 respectively) which regulate at every iteration, the value of e by multiplying either k_1 to increase or k_2 to decrease it. This adjusting procedure depends on the convergence of the real performance value obtained for every iteration. The MATLAB routine *ode23* (low order method) was used to solve the state and co-state differential equations. The iterations were considered to have converged when an increase in the value of J_r was observed. The SIMULINK model was finally used to investigate the final individual performance components. The results are summarised in Table 3.3 and Figures 3.6 to 3.17 (using an 800MHz CPU PC with 256MB RAM).

Case	J_r	Iterat.	J_1	J_2	J_3	J_4	J_5	J
1	601.8149	48	609.34	-7.08×10^{-8}	-3.71×10^{-6}	-7.53	-2.794	-599.01
2	601.9525	253	609.13	-6.96×10^{-8}	-3.09×10^{-6}	-7.18	-6.602	-595.34
3	601.9667	273	609.25	-7.02×10^{-8}	-3.44×10^{-6}	-7.29	-8.831	-593.13

Table 3.3 Performance results from Gradient Method in Function Space

For Case 1, the total time spent for the optimisation was 39.166 seconds. The value of e was fixed to 1.

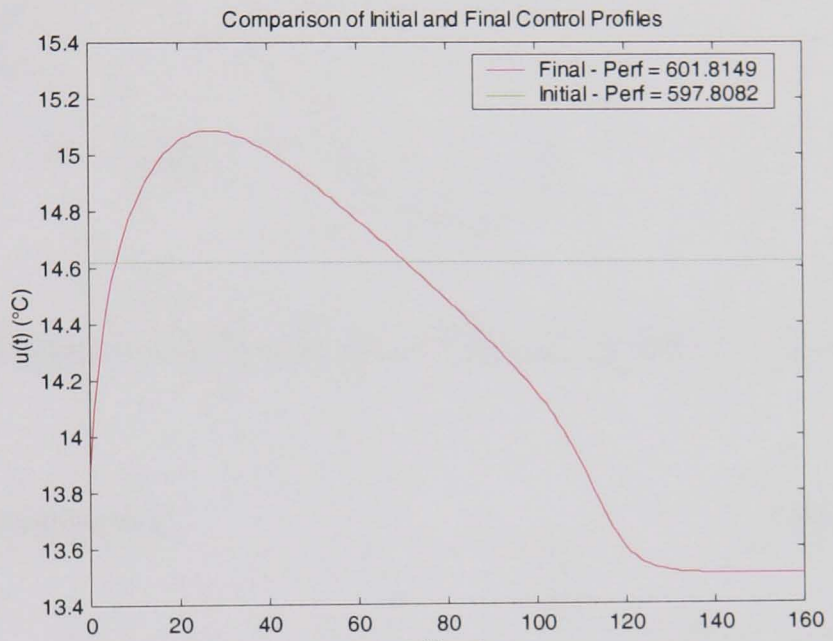

Figure 3.6 Initial and Final Temperature profiles for Case 1

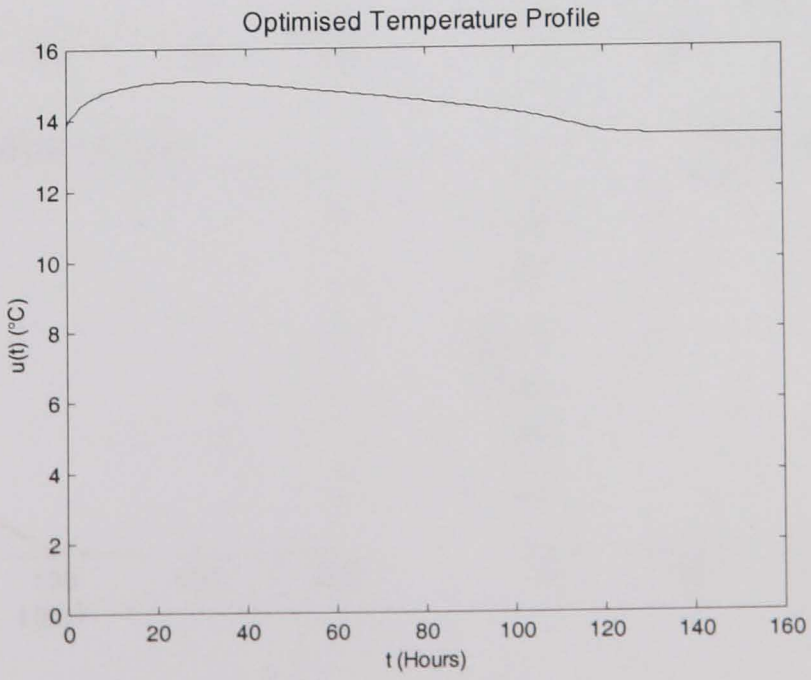

Figure 3.7 Optimised Temperature profile for Case 1 (0 to 16°C scale)

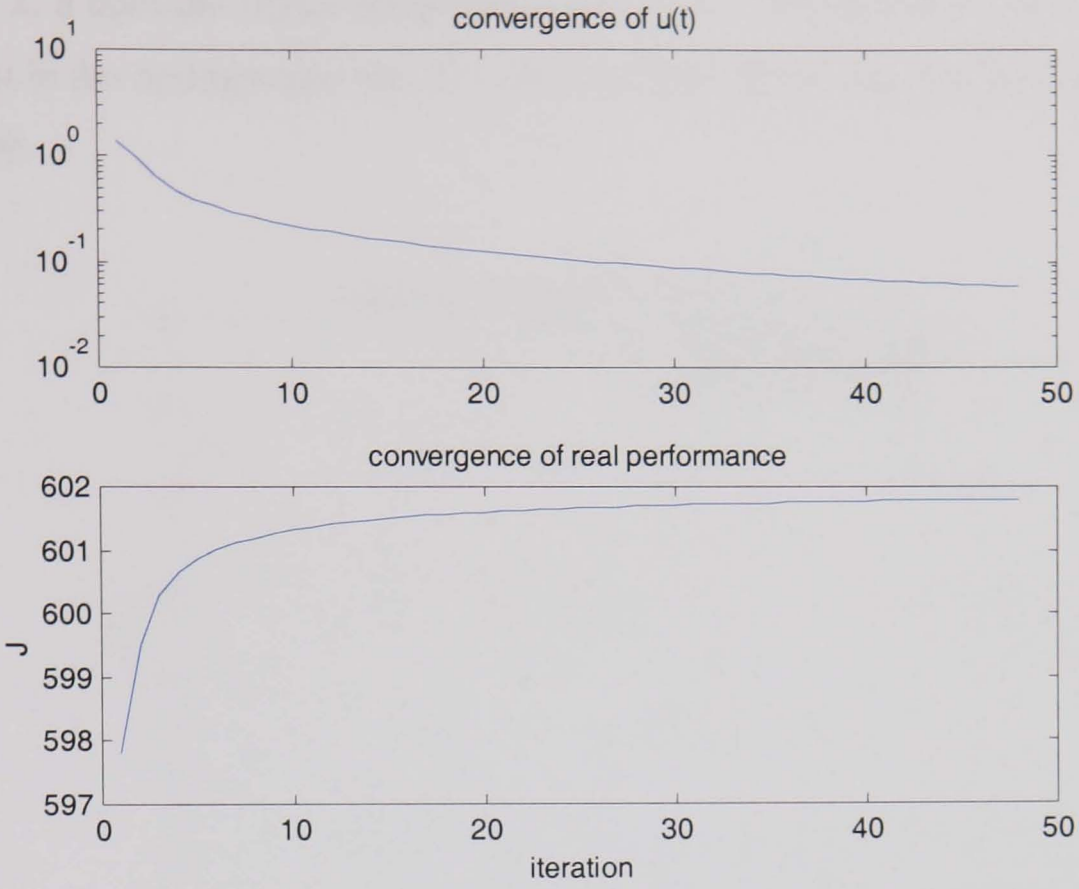

Figure 3.8 Convergence of the parameters for Case 1

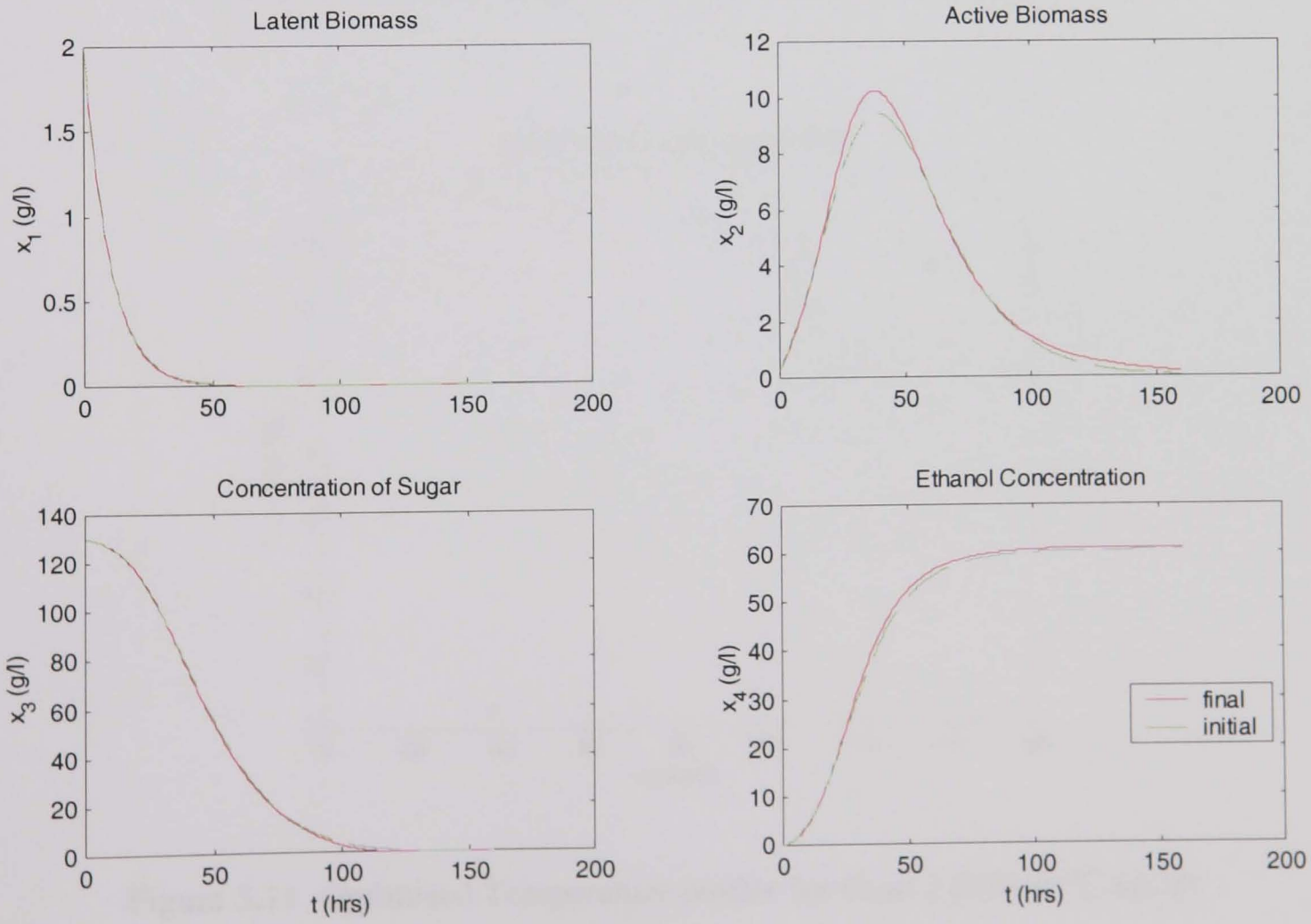

Figure 3.9 Initial and Final state responses for Case 1

For Case 2, a constant initial temperature profile of 10°C has been used. The total time spent in the optimisation was 276.267 seconds. With this, the final value set for *e* was 0.09.

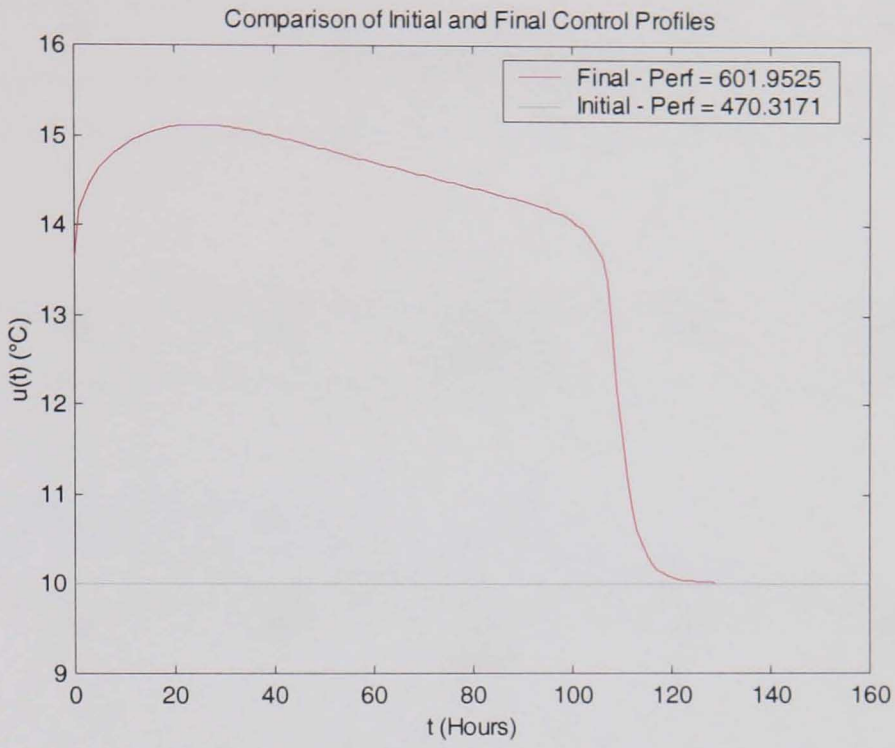

Figure 3.10 Initial and Final Temperature profiles for Case 2

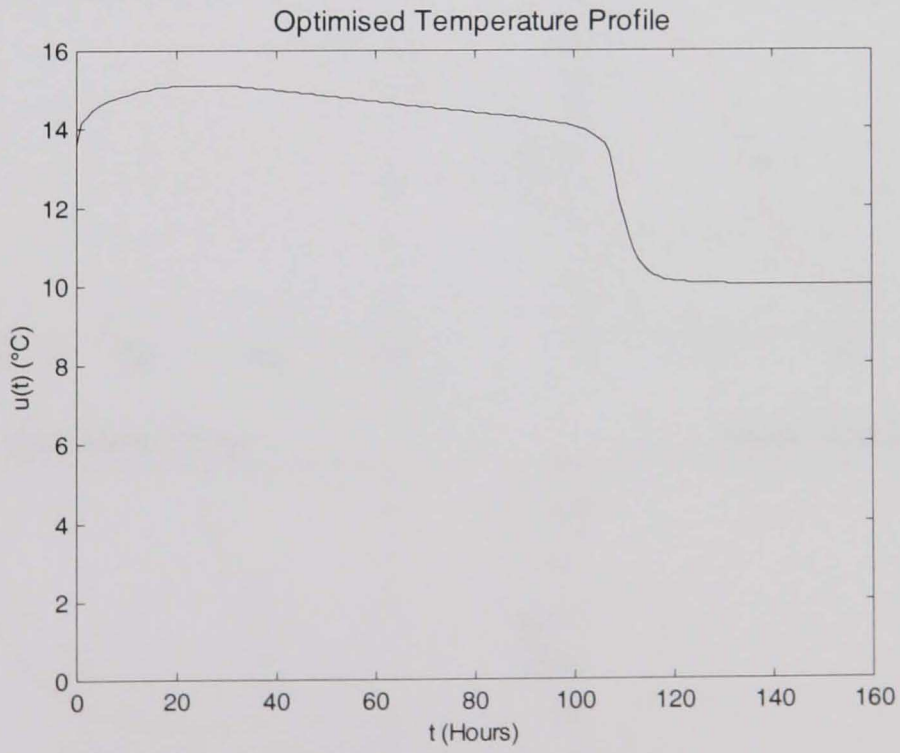

Figure 3.11 Optimised Temperature profile for Case 2 (0 to 16°C scale)

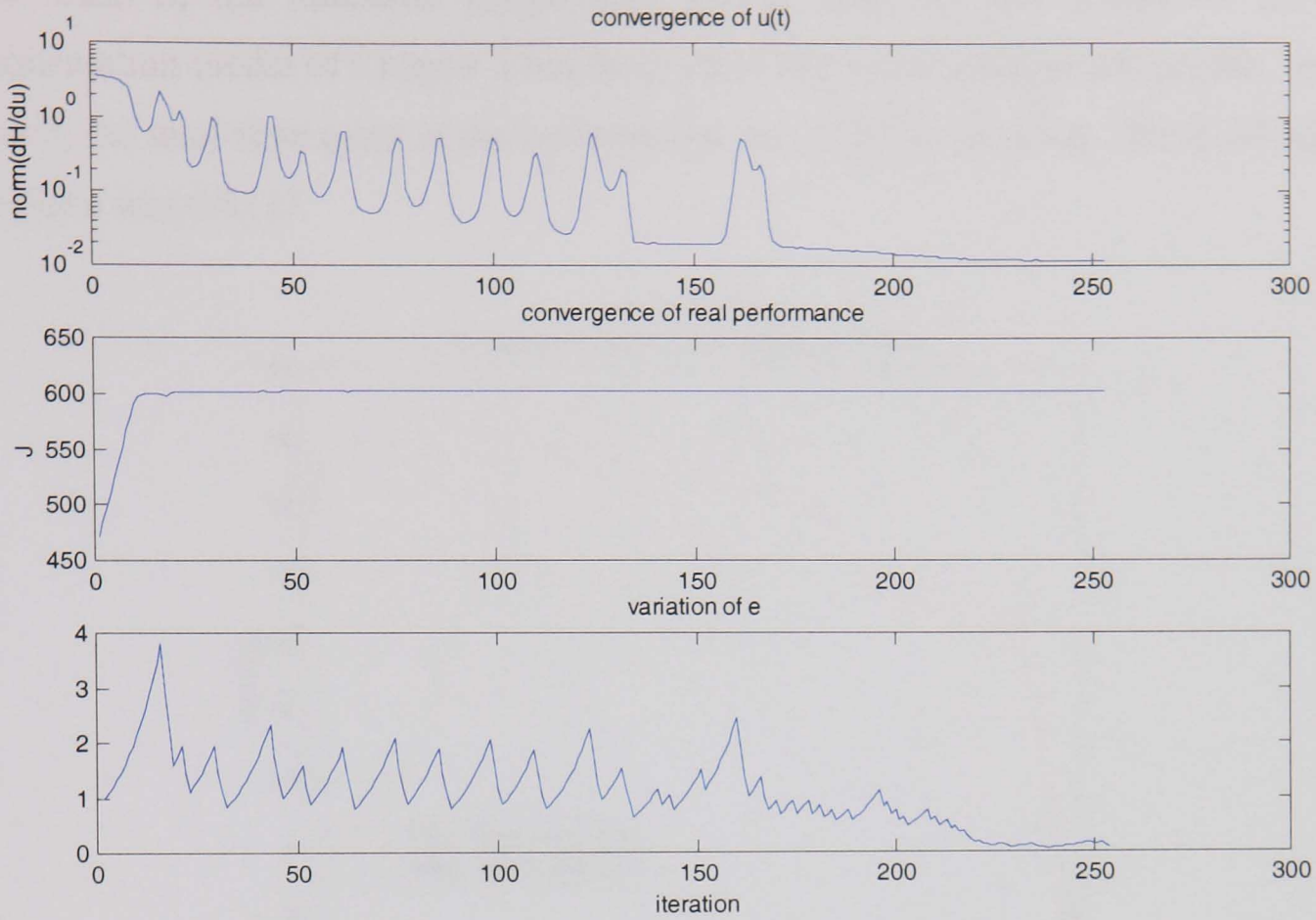

Figure 3.12 Convergence of the parameters for Case 2

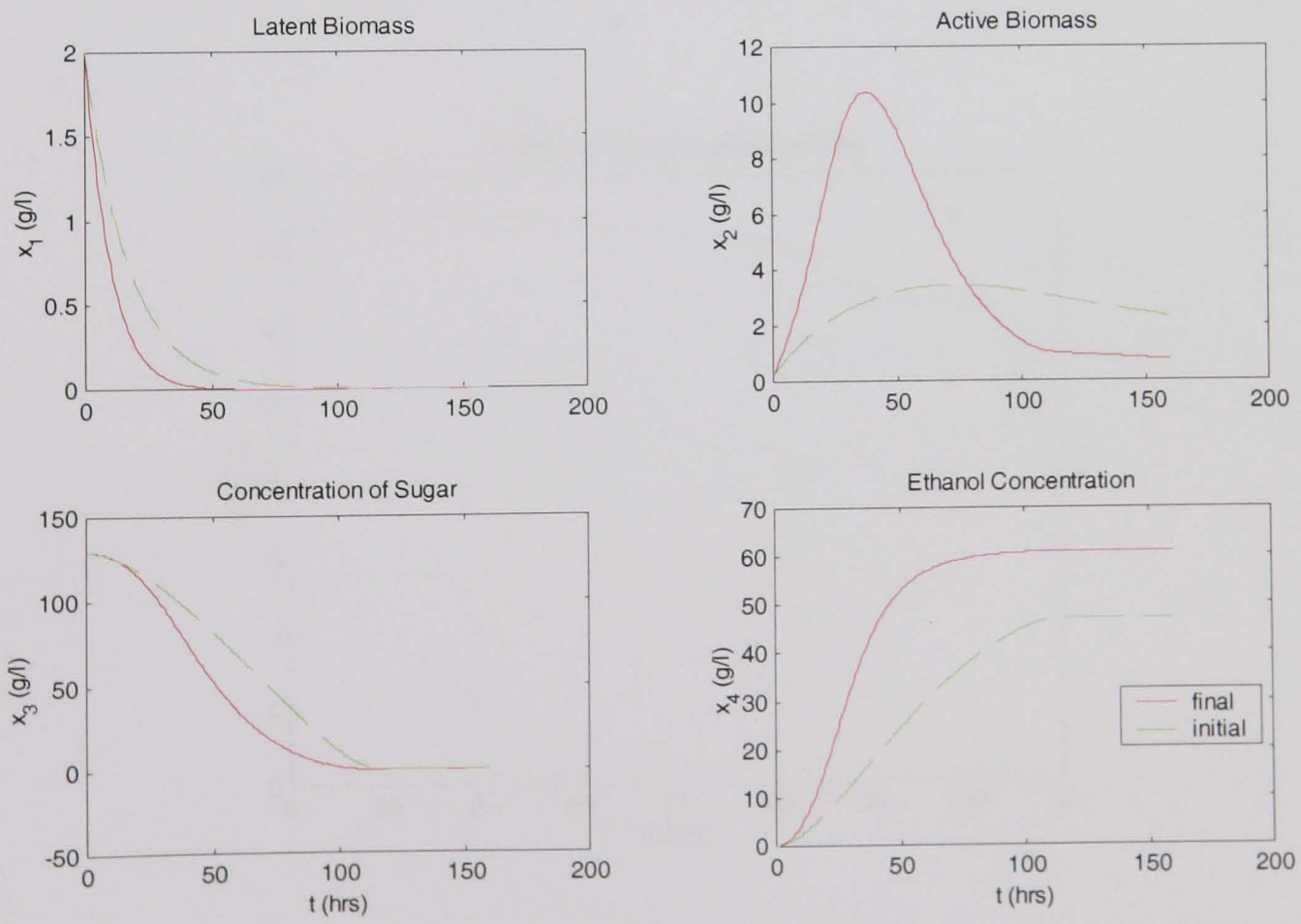

Figure 3.13 Initial and Final state responses for Case 2

For Case 3, the industrial temperature profile used for the simulation of the fermentation model of Chapter 2 has been set as the initial temperature profile. As a result, the total time spent in the optimisation was 278.761 seconds. The final value set for e was 0.4127.

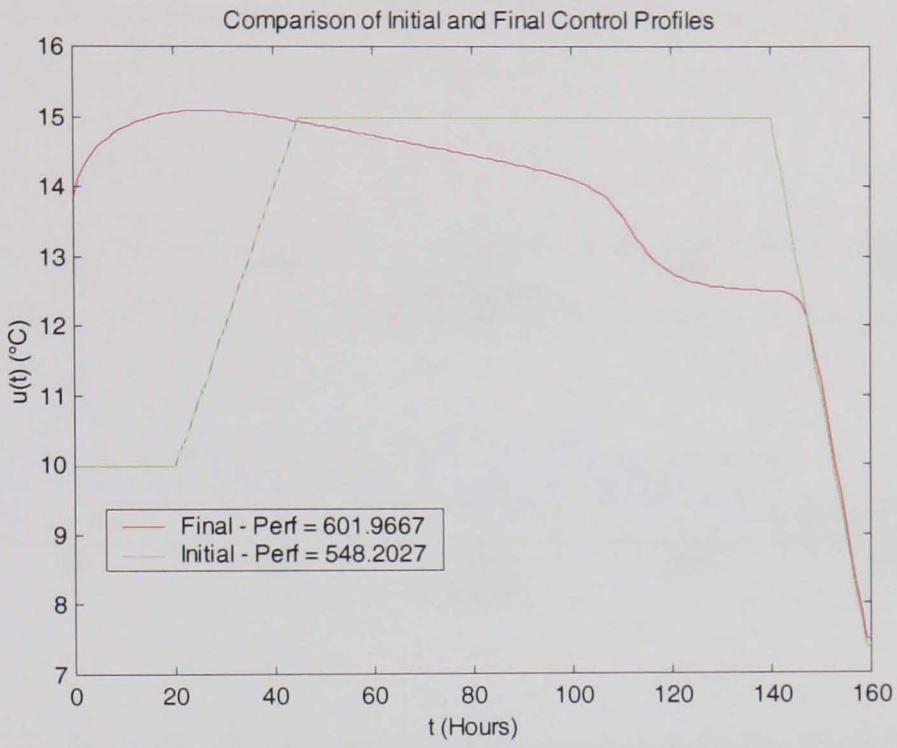

Figure 3.14 Initial and Final Temperature profiles for Case 3

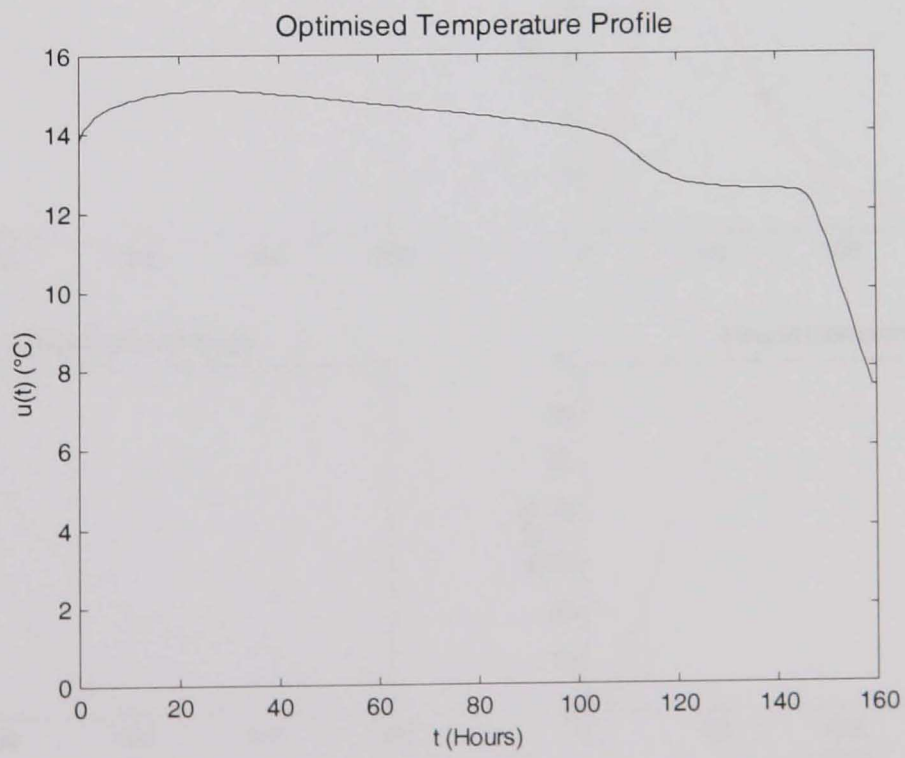

Figure 3.15 Optimised Temperature profile for Case 3 (0 to 16°C scale)

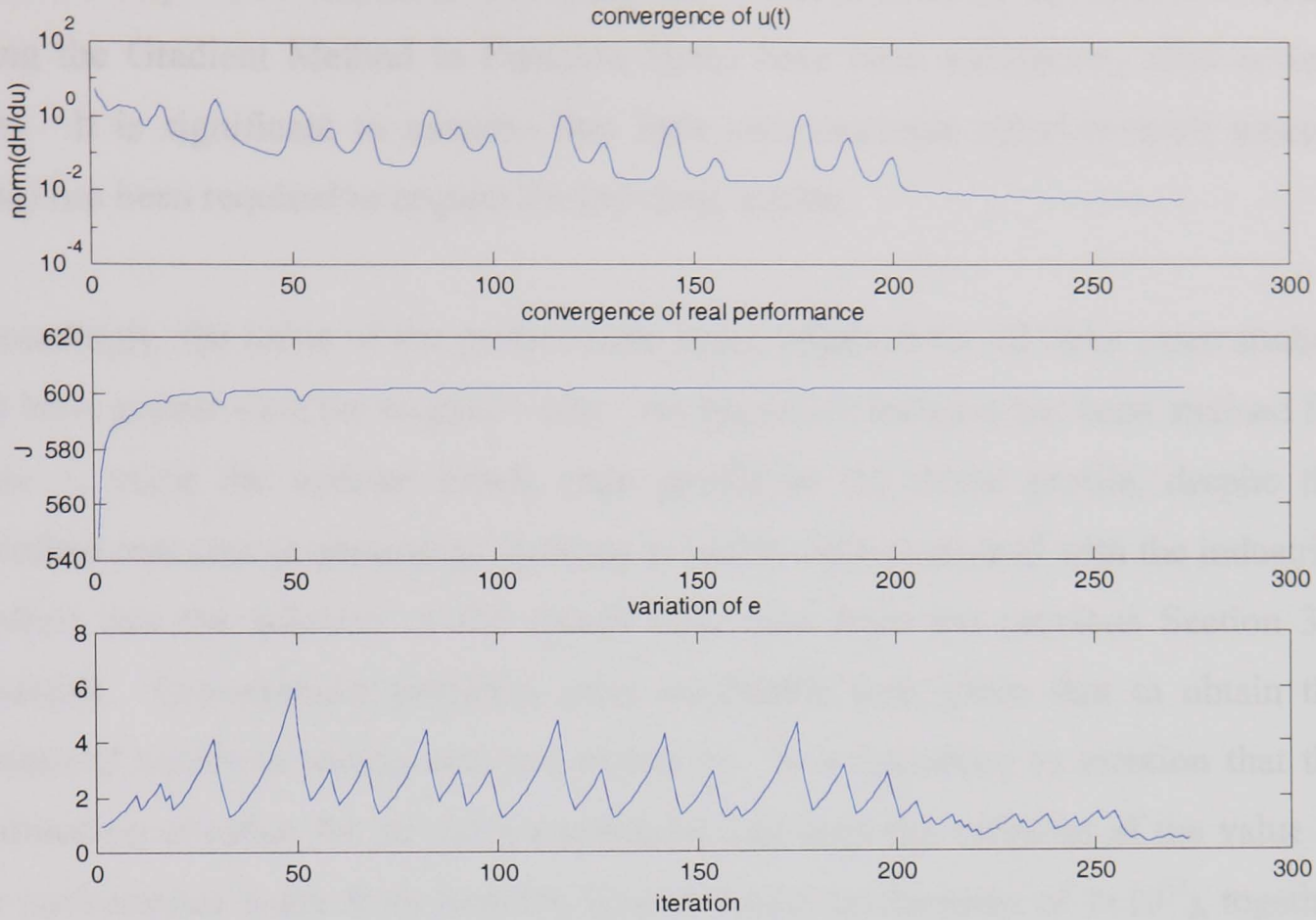

Figure 3.16 Convergence of the parameters for Case 3

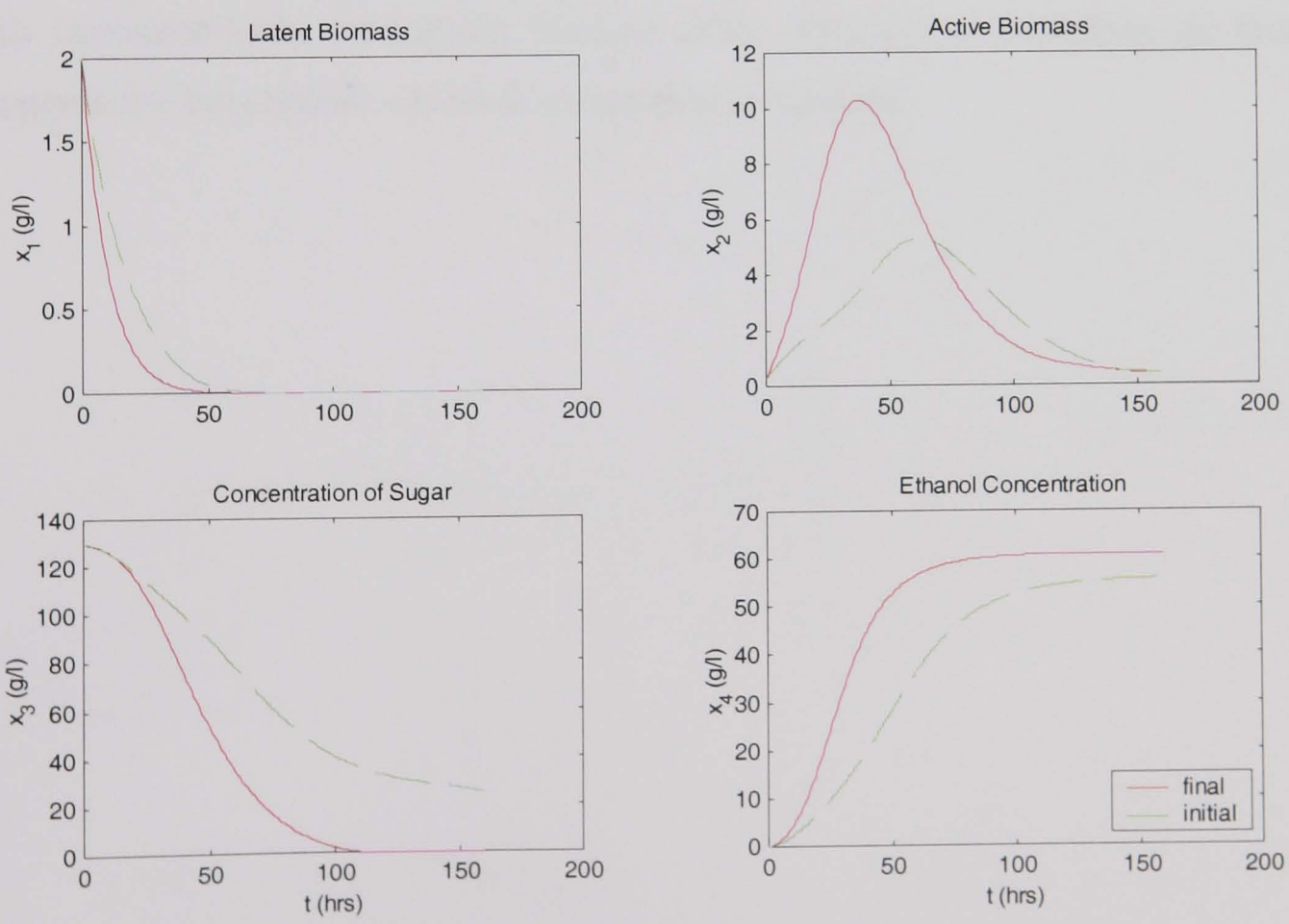

Figure 3.17 Initial and Final state responses for Case 3

With the help of an adaptation technique for the value of e, the optimisation results using the Gradient Method in Function Space have been satisfactory after several tests. It is significant to mention that little computational effort (overall process time) has been required to acquire the previous results.

Accordingly, the value of the performance index obtained for all three cases studied has been greater than the original value. An important increase has been attained for Case 1, using the optimal steady state profile as the initial profile, despite the excellent outcome (a percentage increase of nearly 16% compared with the industrial profile) that the solution of the steady state case from the previous Section 3.2 provided. Convergence problems were reasonably few, given that to obtain the optimised solutions testing was not extensive. It is important to mention that the termination criterion for all cases considered, has been the variation of the value of the performance index from iteration from iteration (in the order of $1x10^{-3}$), together with trying to maintain a smooth optimised temperature profile.

However, it has to be said that the optimised temperature profiles obtained are not easily implemented in the industry because of the difficulty in modelling the brisk changes in the temperature values from iteration to iteration.

3.4 THE DISOPE ALGORITHM

According to Becerra and Roberts (1998a), the DISOPE algorithm is capable of handling nonlinear optimal control problems with terminal state equality consraints. non-quadratic performance indexes, multiple control inputs, control magnitude constraints and continuous time dynamics in an exact way.

In order to apply the DISOPE Algorithm developed by Roberts (1993) for the optimisation of the beer fermentation process; the original principles of the DISOPE algorithm has been taken into consideration in order to derive an appropriate procedure rather than following strictly the modus operandi described earlier in this chapter (this due to the complex nonlinearities of the equations included in the mathematical model). Herewith, the Real Optimal control Problem (ROP) has been considered and it is described from equations 3.48 and 3.49 as:

$$\min_{u(t)} \left\{ \Phi\left(x(t_f)\right) + \int_0^{t_f} ce^{du(t)} dt \right\} \qquad (3.62)$$
$$\text{s.t.:} \quad \dot{x}(t) = f\left(x(t), u(t)\right) \;\; ; \;\; x(0) = x_o$$

Instead of using this ROP, the DISOPE algorithm considers the Model based Optimal control Problem (MOP). This MOP can be described as follows:

$$\min_{u(t)} \left\{ \Phi\left(x(t_f)\right) + \int_0^{t_f} ce^{du(t)} dt \right\} \qquad (3.63)$$
$$\text{s.t.:} \quad \dot{x}(t) = Ax(t) + Bu(t) + \alpha(t) \;\; ; \;\; x(0) = x_o$$

Accordingly, an Expanded Optimal control Problem (EOP) which is equivalent to the ROP can be defined:

$$\min_{u(t)} \left\{ \Phi\left(x(t_f)\right) + \int_0^{t_f} ce^{du(t)} dt \right.$$
$$\text{s.t.:} \quad \dot{x}(t) = Ax(t) + Bu(t) + \alpha(t) \;\; ; \;\; x(0) = x_o$$
$$Az(t) + Bv(t) + \alpha(t) = f\left(z(t), v(t)\right) \qquad (3.64)$$
$$v(t) = u(t)$$
$$z(t) = x(t)$$

The Hamiltonian function $\hat{H}(.)$ has been defined as:

$$\hat{H}(.) = ce^{du(t)} + p^T(t)\big(Ax(t) + Bu(t) + \alpha(t)\big) - \lambda(t)u(t) - \beta^T(t)x(t) \quad (3.65)$$

Then, by means of adjoining the constraints from equation 3.64. the Augmented Performance Index can be obtained:

$$\begin{aligned}\bar{J} &= \Phi\big(x(t_f)\big) \\ &+ \int_{t_o}^{t_f} \Big\{ ce^{du(t)} + p^T(t)\big[Ax(t) + Bu(t) + \alpha(t) - \dot{x}(t)\big] \\ &+ \lambda(t)\big[v(t) - u(t)\big] + \beta^T(t)\big[z(t) - x(t)\big] \\ &+ \mu^T(t)\big[Az(t) + Bv(t) + \alpha(t) - f\big(z(t), v(t)\big)\big]\end{aligned} \quad (3.66)$$

This equation 3.66 can be re-written as:

$$\begin{aligned}\bar{J} &= \Phi\big(x(t_f)\big) \\ &+ \int_0^{t_f} \Big\{ \hat{H}(.) - p^T(t)\dot{x}(t) + \lambda(t)v(t) + \beta^T(t)z(t) \\ &+ \mu^T(t)\big[Az(t) + Bv(t) + \alpha(t) - f\big(z(t), v(t), t\big)\big]\Big\}\, dt\end{aligned} \quad (3.67)$$

At this moment, the assessment of the increment in $\bar{J}$ due to increments in all the variables is pursued. Assuming that the final time is fixed and according to the Lagrange-multiplier theory at a constrained minimum, this increment $\delta\bar{J}$ should be zero (Lewis and Syrmos, 1995). Thus, the 1st variation with fixed initial conditions can be represented as:

$$
\begin{aligned}
\delta \bar{J} = & \left(\nabla_x \Phi(.)\right)^T \delta x\Big|_{t_f} \\
& + \int_0^{t_f} \Big\{ \nabla_u \hat{H}(.)\delta u + \left[\nabla_x \hat{H}(.)\right]^T \delta x \\
& - p^T(t)\delta \dot{x}(t) + \left[\nabla_p \hat{H}(.) - \dot{x}(t)\right]^T \delta p \\
& + \left[\nabla_\alpha \hat{H}(.) + \mu(t)\right]^T \delta\alpha \\
& + \left[\nabla_\lambda \hat{H}(.) + v(t)\right]\delta\lambda + \left[\nabla_\beta \hat{H}(.) + z(t)\right]^T \delta\beta \\
& + \left[\lambda(t) + \left(B - f_v(.)\right)^T \mu(t)\right]\delta v \\
& + \left[\beta(t) + \left(A - f_z(.)\right)^T \mu(t)\right]^T \delta z \\
& + \left[Az(t) + Bv(t) + \alpha(t) - f(.)\right]^T \delta\mu \Big\} dt
\end{aligned}
\tag{3.68}
$$

Now $\int_0^{t_f} p^T(t)\delta\dot{x}dt = p^T(t)\delta x\Big|_{t_f} - p^T(t)\delta x\Big|_{t_0} - \int_0^{t_f} \dot{p}^T(t)\delta x dt$ giving, since for a fixed initial condition $\delta x(0) = 0$, $\int_0^{t_f} p^T(t)\delta\dot{x}dt = p^T(t)\delta x\Big|_{t_f} - \int_0^{t_f} \dot{p}^T(t)\delta x dt$. This gives:

$$
\begin{aligned}
\delta \bar{J} = & \left(\nabla_x \Phi(.) - p(t)\right)^T \delta x\Big|_{t_f} \\
& + \int_0^{t_f} \Big\{ \nabla_u \hat{H}(.)\delta u + \left[\nabla_x \hat{H}_x(.) + \dot{p}(t)\right]^T \delta x \\
& + \left[\nabla_p \hat{H}(.) - \dot{x}(t)\right]^T \delta p \\
& + \left[\nabla_\alpha \hat{H}(.) + \mu(t)\right]^T \delta\alpha \\
& + \left[\nabla_\lambda \hat{H}(.) + v(t)\right]\delta\lambda + \left[\nabla_\beta \hat{H}(.) + z(t)\right]^T \delta\beta \\
& + \left[\lambda(t) + \left(B - f_v(.)\right)^T \mu(t)\right]\delta v \\
& + \left[\beta(t) + \left(A - f_z(.)\right)^T \mu(t)\right]^T \delta z \\
& + \left[Az(t) + Bv(t) + \alpha(t) - f(.)\right]^T \delta\mu \Big\} dt
\end{aligned}
\tag{3.69}
$$

Consequently, for $\delta\bar{J} = 0$ the following necessary optimality conditions can be attained:

$$\left\{\begin{aligned}\nabla_u \hat{H}(.) &= 0 \\ \dot{x}(t) &= \nabla_p \hat{H}(.) \\ \dot{p}(t) &= -\nabla_x \hat{H}(.)\end{aligned}\right\} \tag{3.70}$$

$$\left[\nabla_x \Phi(.) - p(t_f)\right]^T \delta x(t_f) = 0 \tag{3.71}$$

$$Az(t) + Bv(t) + \alpha(t) - f\left(z(t), v(t), t\right) = 0 \tag{3.72}$$

$$\left\{\begin{aligned}\lambda(t) &= \left(B - f_v(.)\right)^T p(t) \\ \beta(t) &= \left(A - f_z(.)\right)^T p(t)\end{aligned}\right\} \tag{3.73}$$

$$\left\{\begin{aligned}v(t) &= u(t) \\ z(t) &= x(t)\end{aligned}\right\} \tag{3.74}$$

Thus, the optimal control is given from the Hamiltonian gradient:

$$H_u = cde^{du(t)} + B^T p(t) - \lambda(t) = 0 \tag{3.75}$$

giving

$$u(t) = \frac{1}{d}\log_e\left(\frac{\lambda(t) - B^T p(t)}{cd}\right) \tag{3.76}$$

which, noting c and d are positive, has a real solution, called the validity solution, if:

$$\lambda(t) - B^T p(t) > 0 \tag{3.77}$$

Note that if the previous inequality 3.77 is not satisfied, applying the Minimum Principle to minimise the Hamiltonian of equation 3.65 requires $u(t)$ to be chosen to minimise:

$$\bar{H}(.) = ce^{du(t)} + \left(p^T(t)B - \lambda(t)\right)u(t);\ p^T(t)B - \lambda(t) \geq 0 \tag{3.78}$$

which is minimised when *u(t)* takes on its smallest value, that is when:

$$u(t) = u_{\min} \tag{3.79}$$

Hence, the optimal control is:

$$u(t) = \begin{cases} \dfrac{1}{d}\log_e\left(\dfrac{\lambda(t) - B^T p(t)}{cd}\right) & \text{if } \lambda(t) - B^T p(t) > 0 \\ u_{\min} & \text{if } \lambda(t) - B^T p(t) \leq 0 \end{cases} \tag{3.80}$$

The *two point boundary* value problem is then:

$$\dot{x}(t) = x(t) + Bu(t) + \alpha(t)\ ,\ x(0) = x_o \tag{3.81}$$

$$\dot{p}(t) = -A^T p(t) + \beta(t)\ ,\ p(t_f) = \nabla_x \Phi(.) \tag{3.82}$$

It is important to observe that the co-state equation can be solved independently of *x(t)* and *u(t)*. In addition, the state equations and the co-state equations can be solved analytically because they are both linear. This gives computational advantages over the gradient method in function space technique described in the previous Section 3.3. However, the optimal control given by equation 3.80 introduces a discontinuity.

The following algorithm reduces the effect of the discontinuity by employing a gradient method to apply a correction to each iteration of control *u(t)* along the portion of the profile when the inequality 3.77 is not satisfied. Then, a correction to the final solution is applied using equation 3.80. This is a batch type algorithm (Roberts and Becerra, 1997).

The algorithm can be described as follows:

Data: t_f, e, k_u, k_p, $u_{\min}$, A, B.

Step 0: *initialisation*: Set iteration counter $k = 1$, and guess $v(t)^0$ for the interval $[0, t_f]$. Also find $p(t)^0$ by solving equation 3.82 backwards in time from $t = t_f$ with $\beta(t) = 0$. Set $\hat{p}(t)^0 = p(t)^0$.

Step 1: *application to reality*: Compute $z(t)$ and J_r from equations 3.49 and 3.48 with $z(t)$ replacing $x(t)$, $v(t)^k$ replacing $u(t)$.

Step 2: *parameter estimation*: Compute $\alpha(t)$ from equation 3.72.

Step 3: *modifiers*: Compute $\lambda(t)$ and $\beta(t)$ from equation Set 3.73 with $p(t) = \hat{p}(t)^k$.

Step 4: *optimisation* (*two-point boundary value problem*):

(i) Compute the co-state vector $p(t)$ by solving equation 3.82 in reverse time. Then compute the Hamiltonian gradient H_u from equation 3.75 with $v(t)^k$ replacing $u(t)$.

(ii) Over the interval $[0, t_f]$, when the inequality 3.77 is satisfied, use equation 3.76 to compute the predicted optimal control $u(t)$; when the inequality is not satisfied compute $u(t)$ by the gradient technique:

$$u(t) = v(t)^k - eH_u \tag{3.83}$$

(iii) Compute $x(t)$ using equation 3.81.

Step 5: *update*: Use the following relaxation technique to update $v(t)$ and $\hat{p}(t)$

$$\left.\begin{aligned} v(t)^{k+1} &= v(t)^k + k_u\left(u(t) - v(t)^k\right) \\ \hat{p}(t)^{k+1} &= \hat{p}(t)^k + k_p\left(p(t) - \hat{p}(t)^k\right) \end{aligned}\right\} \tag{3.84}$$

Set $k = k+1$ and repeat from **Step 1** until convergence is achieved.

Step 6: *after convergence*: Correct the final solution for $u(t)$ using equation 3.80.

The subsequent three cases have been taken into consideration:

Case A: $u(t)$ has been initialised at the optimum steady-state solution:

$$u(t) = 14.6211,\ t \in [0,160]$$

Case B: $u(t)$ has been initialised at an arbitrary steady-state solution:

$$u(t) = 10,\ t \in [0,160]$$

Case C: $u(t)$ has been initialised at the industrial profile.

The MATLAB implementation used *ode23* for solving the differential equations in Step 1, and the linear differential equation solver *lsim* in Steps 0, 4 (i) and 4 (iii). In all three cases, the Jacobian matrices A and B were determined by linearising about a nominal solution $\bar{u}=14$, $\bar{x}=\begin{bmatrix}0.2 & 1 & 10 & 50\end{bmatrix}^T$ eventhough they can also be obtained with the help of the System Identification Toolbox from MATLAB as it is shown in Appendix B. This due to the fact that the results obtained with both of these techniques have been very similar, and the linear differential equation solver method can be applied within the MATLAB script file. An additional novel approach for System Identification can be found in Appendix C, this time using Neural Networks.

The termination criterion for all cases considered with the DISOPE algorithm has been again the variation in the value of the performance index from one iteration to another (in the order of $1\text{x}10^{-3}$). The results are summarised in Table 3.4 and Figures 3.18 to 3.29 (using an 800MHz CPU PC with 256MB RAM).

Case	J_r	Iterat.	J_1	J_2	J_3	J_4	J_5	J
A	600.6426	65	606.96	-7.28×10^{-8}	-3.65×10^{-7}	-6.31	-1.78	598.87
B	601.9361	444	608.54	-7.02×10^{-8}	-1.72×10^{-6}	-6.61	-6.809	595.12
C	601.5659	175	607.95	-7.55×10^{-8}	-1.00×10^{-6}	-6.44	-6.219	595.29

Table 3.4 Performance results from the DISOPE Algorithm

In Case A, an initial profile $u(t)^0$ set at the optimal steady-state profile was considered. The subsequent results were obtained using these set values: e=0.5, k_u=0.0075, k_p=0.1, $u_{\min}$=0, $u_{\max}$=20. Convergence was achieved after just 65 iterations in a total time of 41.429 seconds with final J_r=600.6426.

The final corrected control profile is shown in Fig. 3.18 and 3.19 (full scale). Fig. 3.20 shows the convergence of $u(t)$ for every iteration and also the variation in the performance index. The corresponding state response is given in Fig. 3.21.

Furthermore, it was observed that each iteration of the DISOPE algorithm took less time that a single iteration of the Gradient Algorithm according to the ratio 1/1.29.

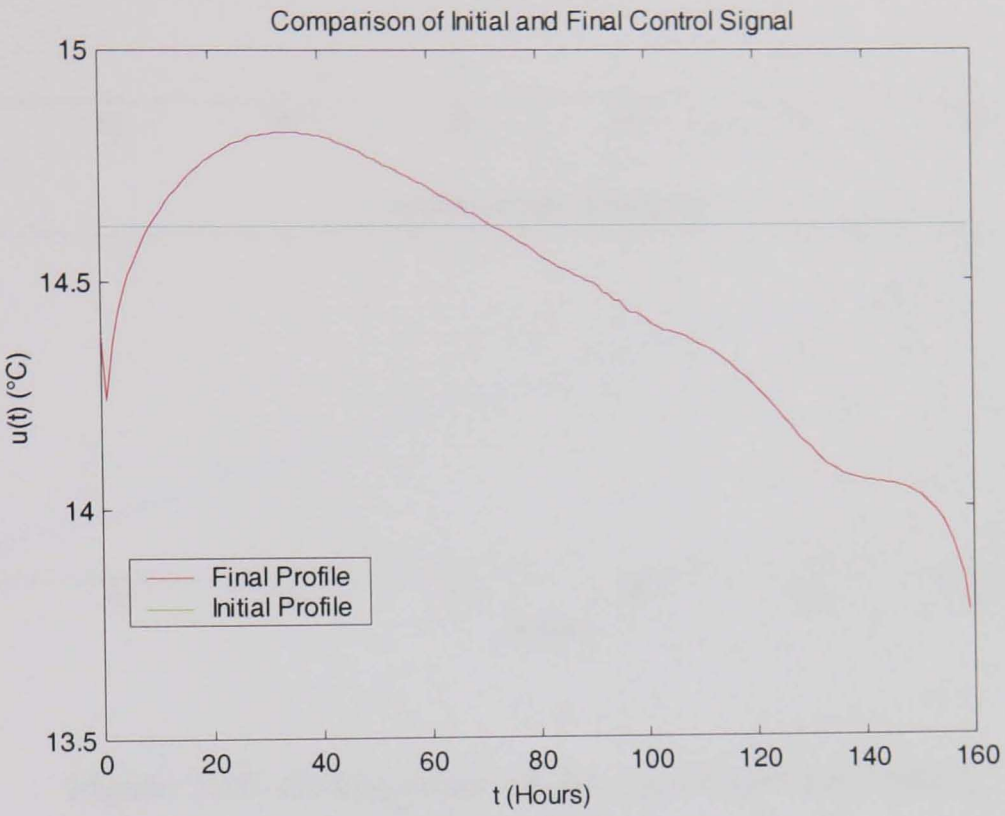

Figure 3.18 Initial and Final Temperature profiles for Case A

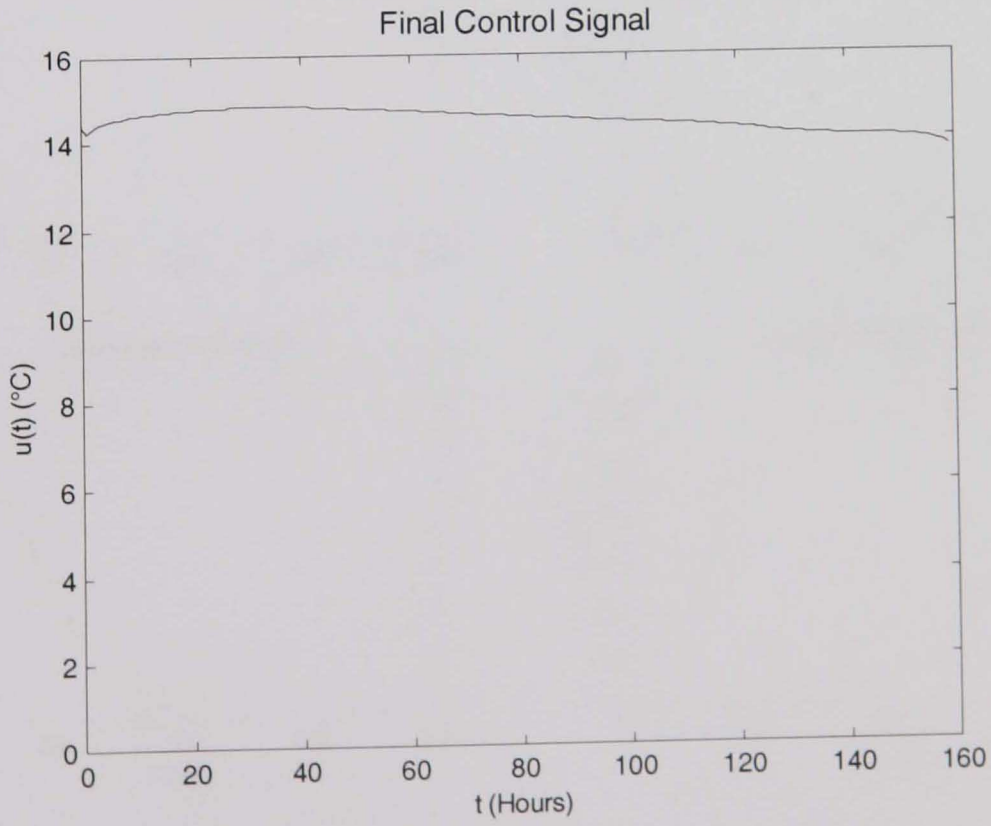

Figure 3.19 Optimised Temperature profile for Case A (0 to 16°C scale)

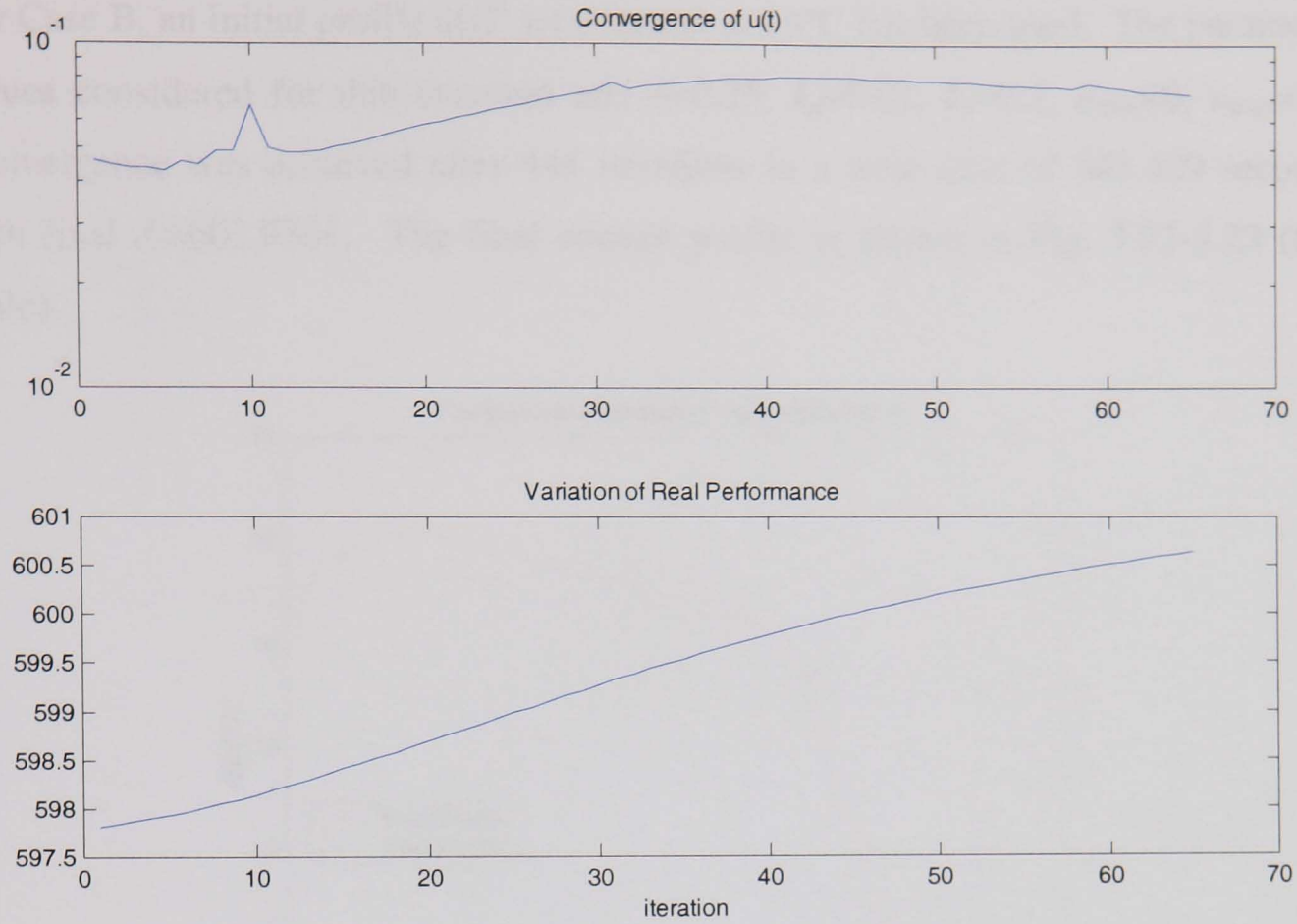

Figure 3.20 Convergence of the parameters for Case A

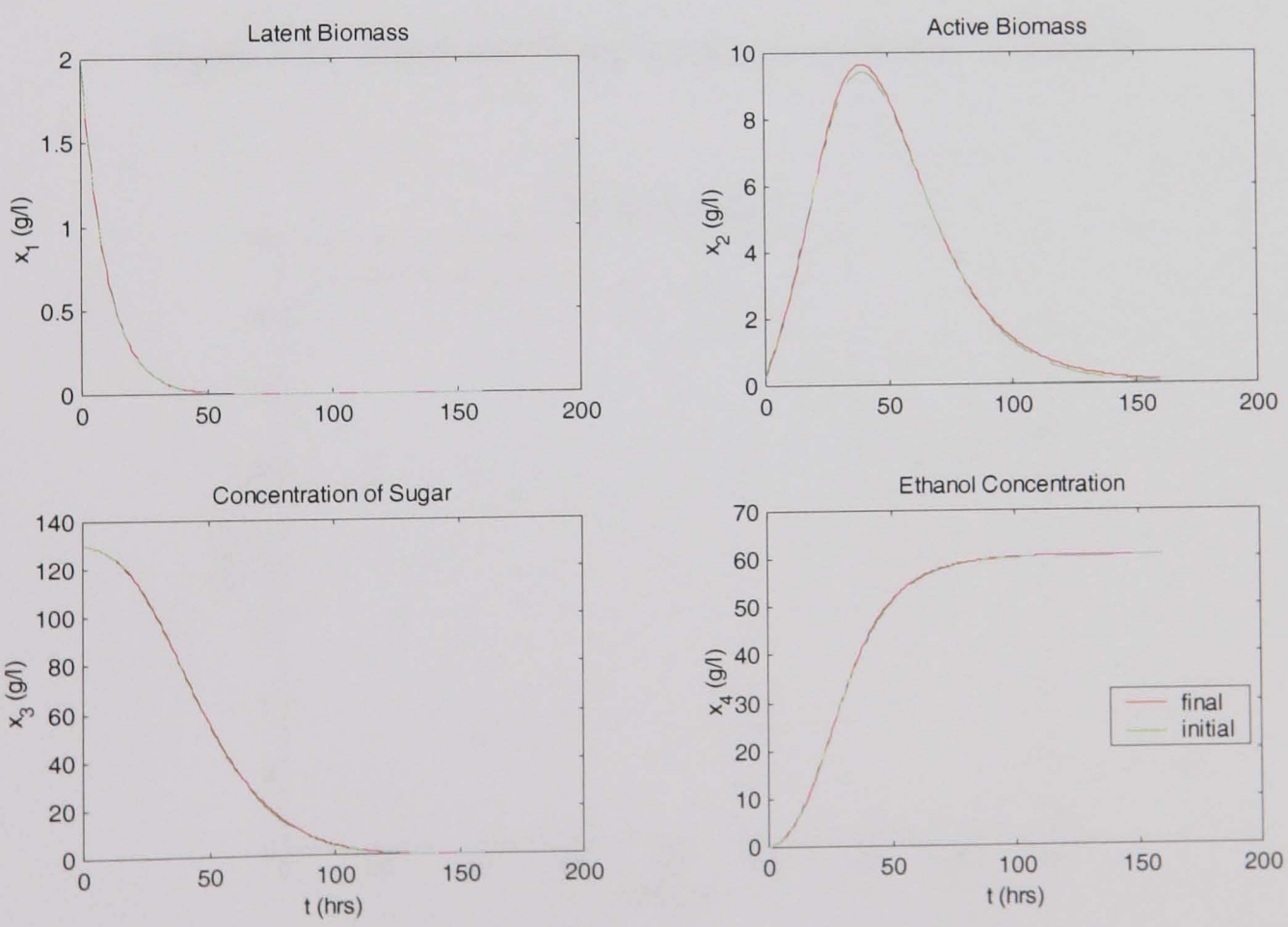

Figure 3.21 Initial and Final state responses for Case A

For Case B, an initial profile $u(t)^0$ set constant at 10°C has been used. The parameter values considered for this situation are: e=0.25, k_u=0.01, k_p=0.2, u_{min}=0, u_{max}=20. Convergence was achieved after 444 iterations in a total time of 341.429 seconds with final J_r=601.9361. The final control profile is shown in Fig. 3.22-3.23 (full scale).

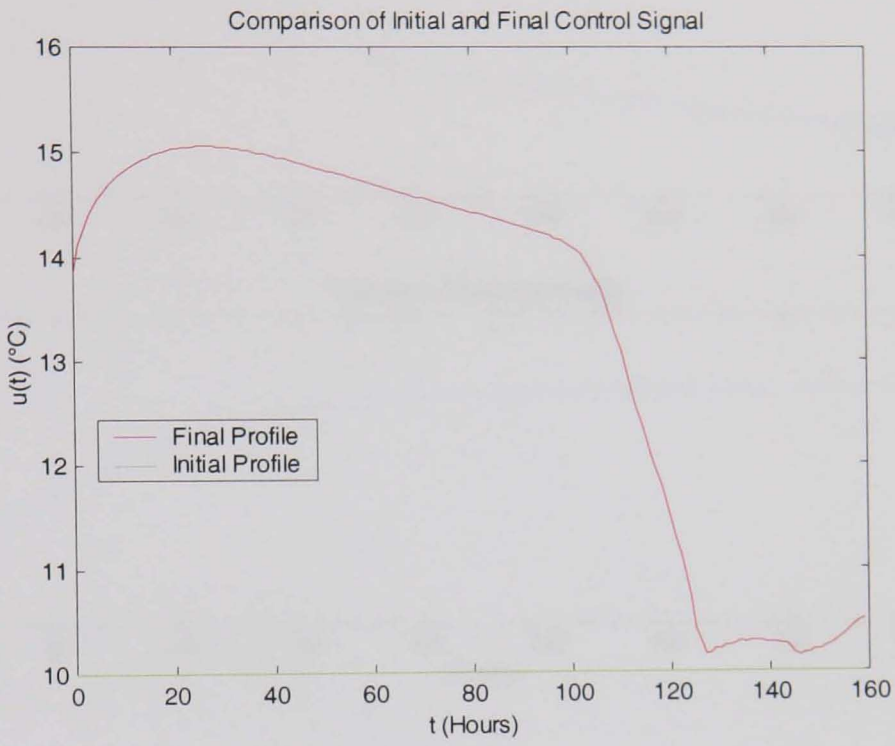

Figure 3.22 Initial and Final Temperature profiles for Case B

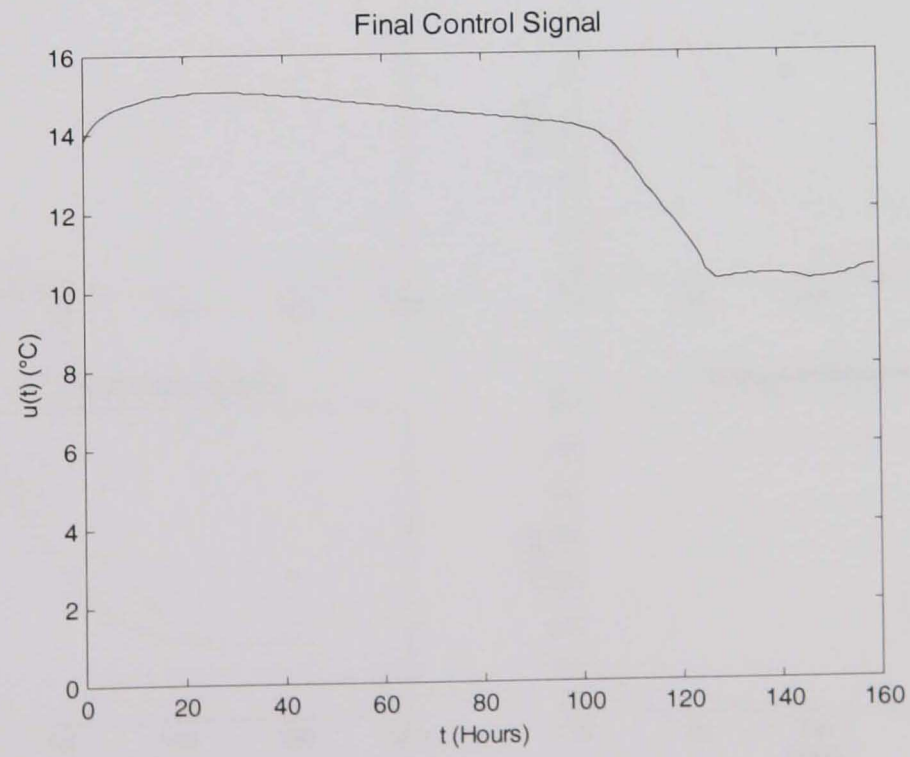

Figure 3.23 Optimised Temperature profile for Case B (0 to 16°C scale)

Fig. 3.24 shows the convergence of *u(t)* and the variation in the performance index along the optimisation process. The corresponding state response is given in Fig. 3.25.

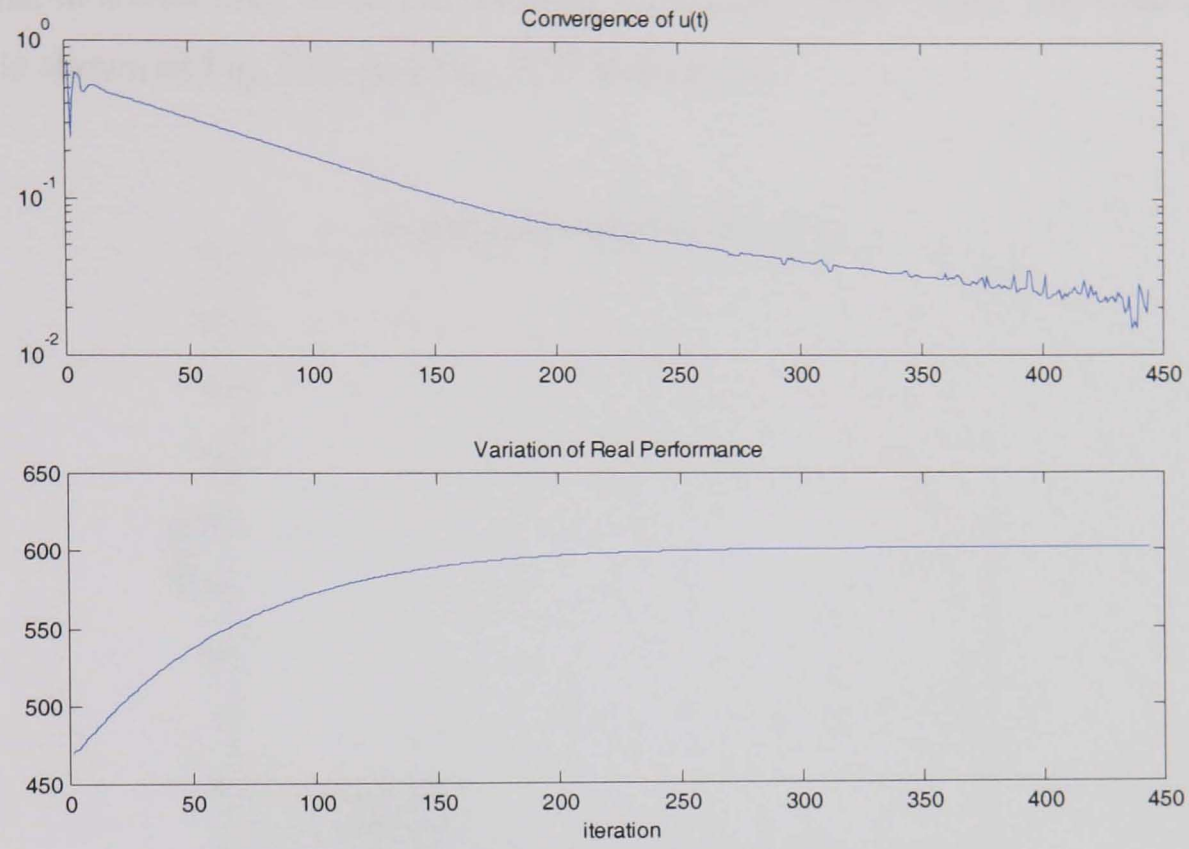

Figure 3.24 Convergence of the parameters for Case B

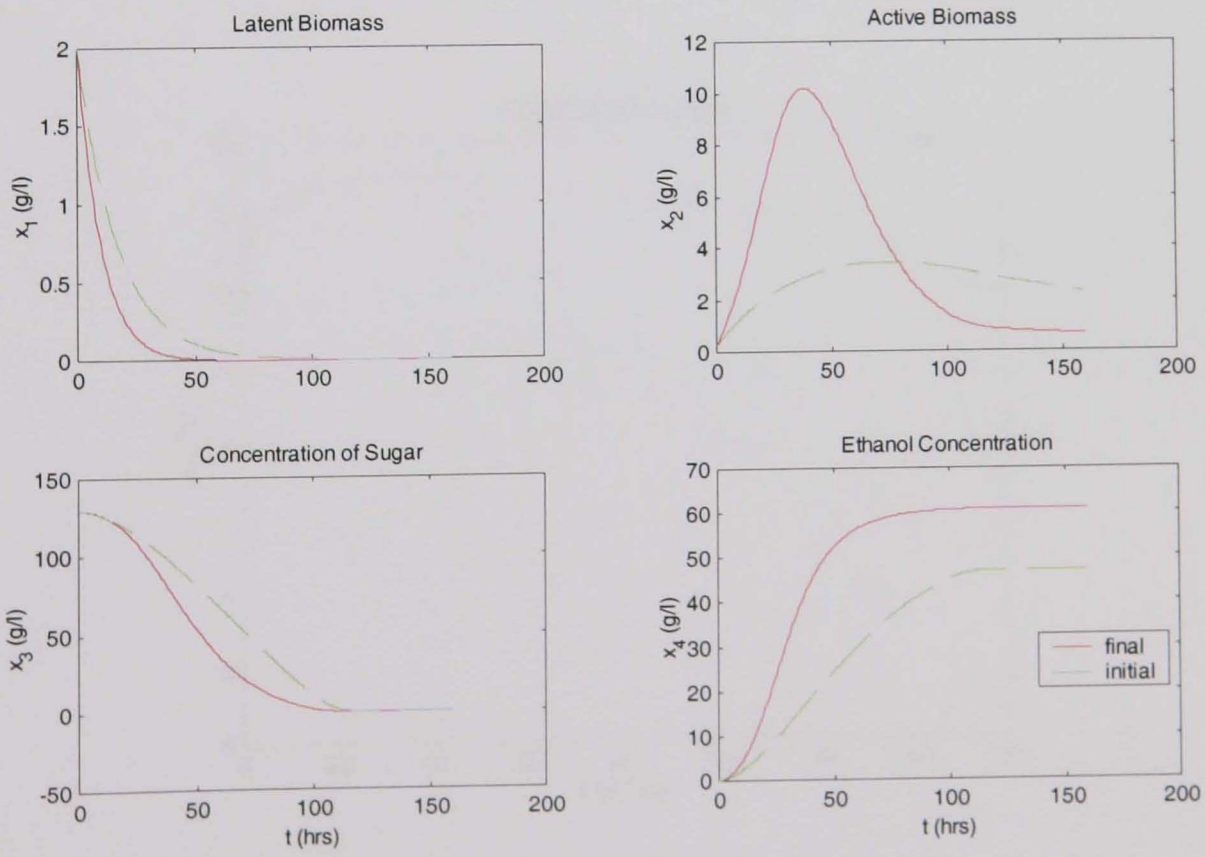

Figure 3.25 Initial and Final state responses for Case B

The temperature profile used by the industry has been selected as the initial profile $u(t)^0$ for Case C. The parameter values considered for this situation: e=2, k_u=0.02, k_p=0.5, u_{min}=0, u_{max}=20. An appropriate optimised profile was achieved in 175 iterations, in a total time of 92.122 seconds with final J_f=601.5659. The final control profile is shown in Fig. 3.26 and Fig. 3.27 (full scale).

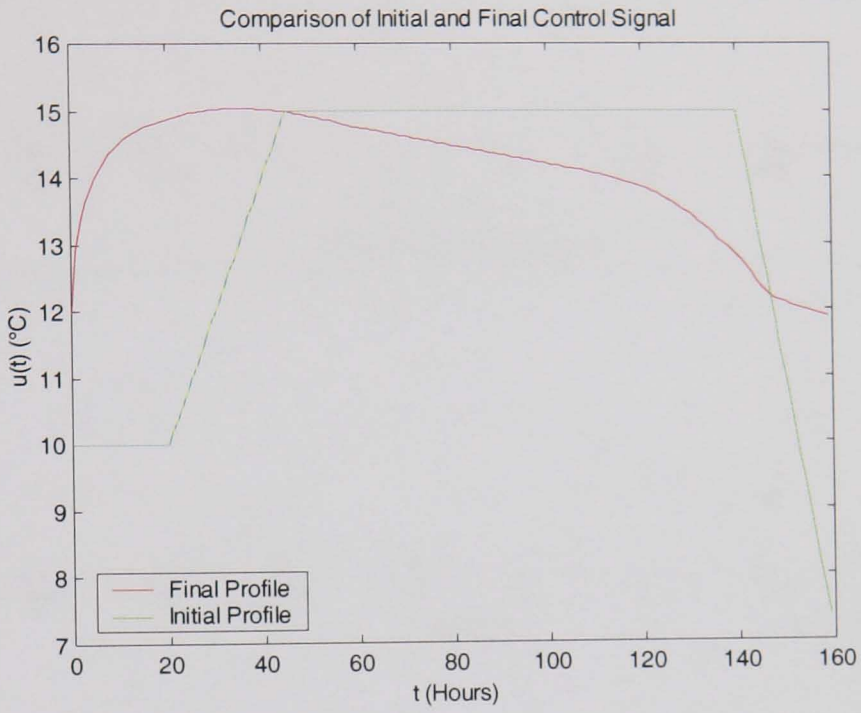

Figure 3.26 Initial and Final Temperature profiles for Case C

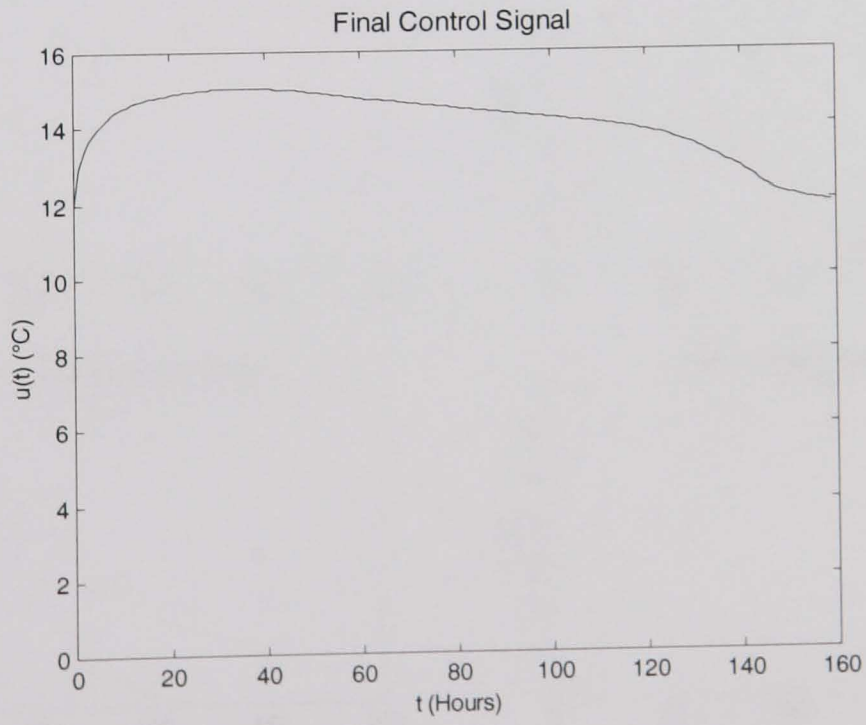

Figure 3.27 Optimised Temperature profile for Case C (0 to 16°C scale)

Fig. 3.28 shows the convergence of *u(t)* and the variation in the performance index along the optimisation process. The corresponding state response is given in Fig. 3.29.

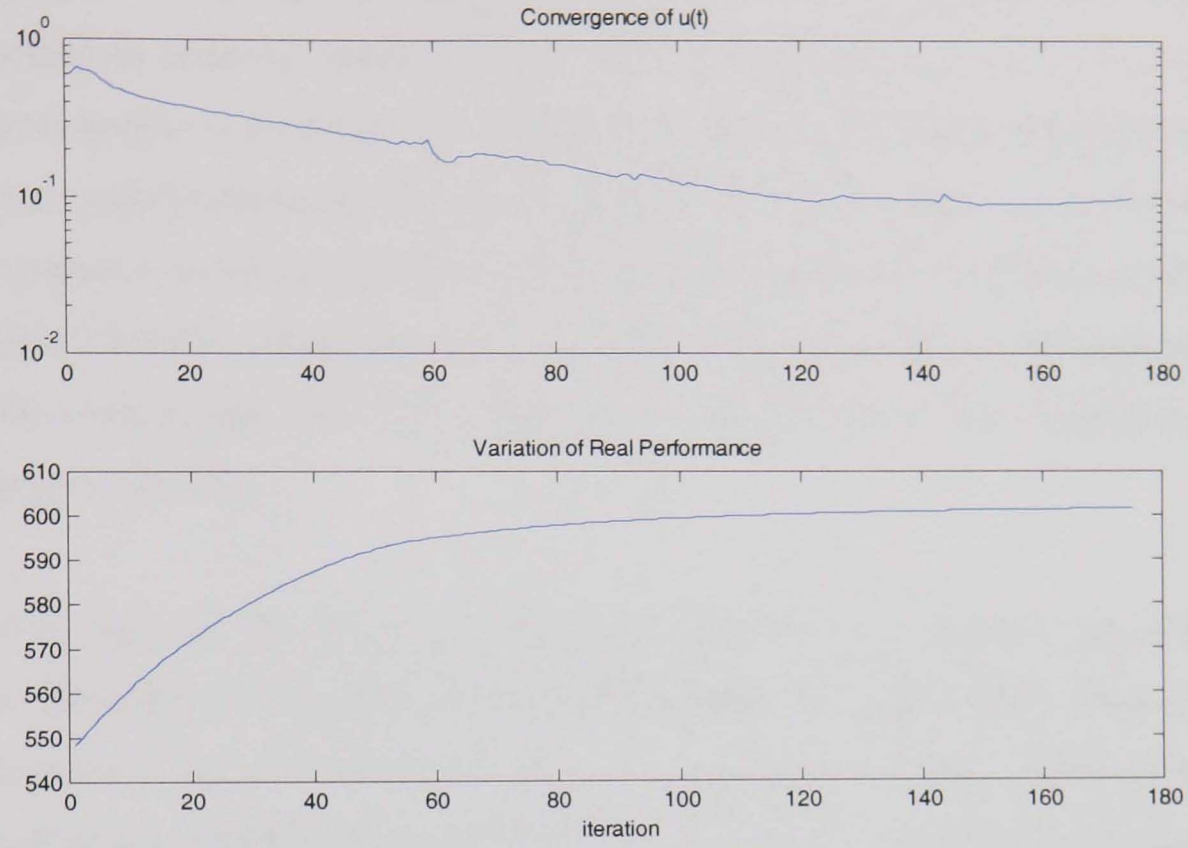

Figure 3.28 Convergence of the parameters for Case C

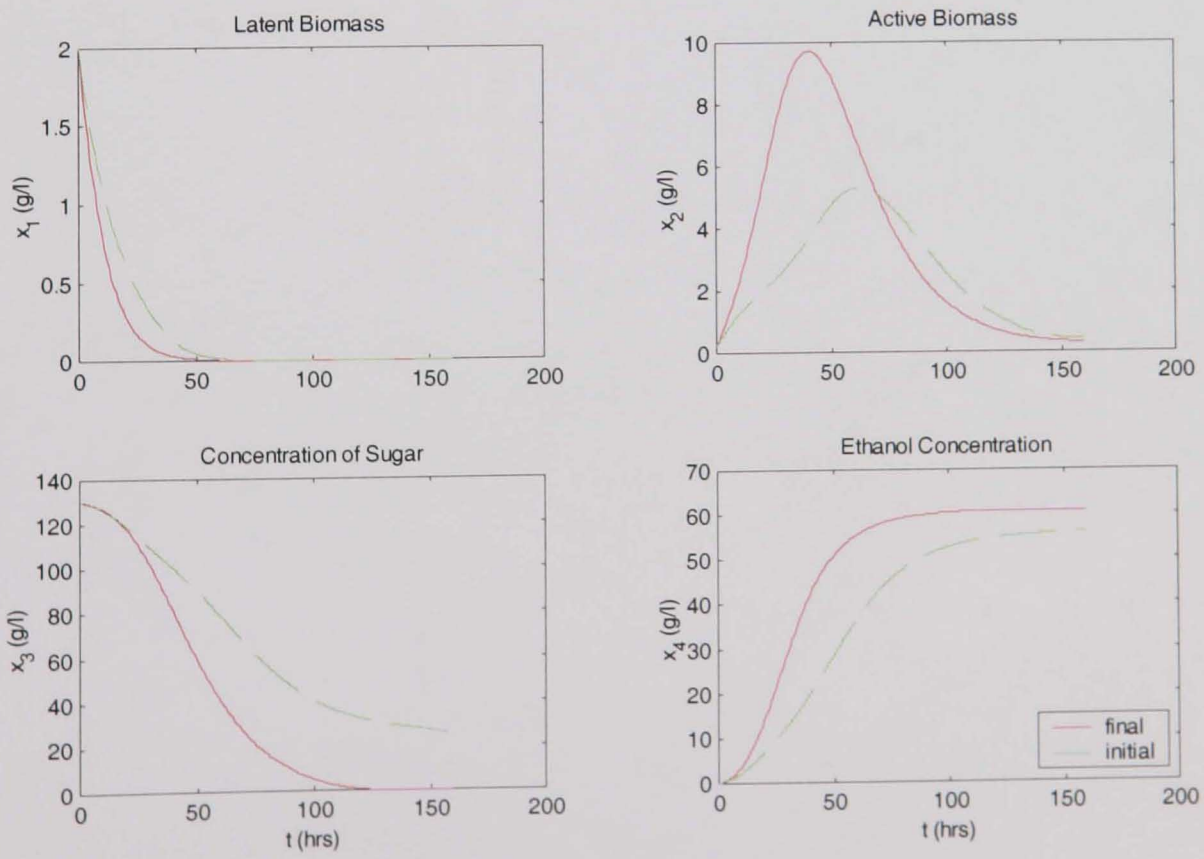

Figure 3.29 Initial and Final state responses for Case C

The results obtained with the batch version of the DISOPE algorithm are considered to be very important for optimising the simulated model. Nevertheless, further studies need to be carried out in order to see the viability of this algorithm with this kind of process. Convergence problems were noticed as large number of tests had to be performed in order to obtain the right initial parameters (e, k_u, k_p) for a suitable temperature profile to be achieved. Additionally, for the best case among all the tests carried out, a performance index value of just 598.87 was obtained: which is slightly less if compared to the one obtained for the best profile using the Gradient Method. All of these should be taken into account when other optimisation methods become part of the research since the main objective is to obtain the highest possible value in this performance index.

In addition, although the results obtained show some improvement in the objective function, industrial application will be rather difficult since the temperature profiles are inappropriate for practical implementation (breweries normally use profiles with few temperature changes along the fermentation time). Obtaining implementable profiles is going to be an essential aspect of the research to be carried out in this work.

3.5 SUMMARY

This chapter comprises an attempt to optimise the mathematical model of the beer fermentation process selected from the work done by Andres-Toro et al. (1998) and reviewed in the second chapter. In order to try to obtain similar responses using the same model, some parameters have been changed and adapted so they follow the industry's behaviour. A representation of the SIMULINK version of the process modelled is included in this chapter; this model incorporates an M-File (S-Function) that comprises the differential equations as well as the initial values of the mathematical model.

The SIMULINK model itself has been adapted to evaluate the objective function J in order to identify the initial value to be optimise (for a 160 hours fermentation time); this was done taking into account the original objective function developed in the original work.

An introduction to the DISOPE algorithm developed by Roberts (1993) and later revised by Becerra and Roberts (1998a) has been included in the first section. The fermentation process itself has been reviewed in detail in the next section and this includes also some special considerations that have been made to it. The optimal steady state solution can be considered the primary effort to optimised the process and it is a good starting point for the optimisation techniques to follow.

After the application of the Minimum Principle, a solution using a Gradient Method in Function Space has been achieved. Results for three particular cases has been included and illustrate in some detail how good this optimisation technique is for the fermentation process considered.

With the help of the continuous version of the DISOPE algorithm (modified from original work by Roberts, 1993) and some other considerations made to adjust it to the simulated process; the optimisation of the fermentation process has also been accomplished. The maximisation of the objective function for three cases has been presented and it also includes some comparison notes with previous sections.

CHAPTER IV

GENETIC ALGORITHMS

This chapter introduces Genetic Algorithms as an optimisation technique for non-linear processes. A brief preface on the history and development of Genetic Algorithms in the field of Evolutionary Computation is reviewed; emphasis has been made in optimal control applications. The relevance of random search techniques finding the global optimum for non-linear problems without the need of an initial guess has been discussed. The Genetic Algorithms Toolbox by Chipperfield et al (1994) has been used for the optimisation of the beer fermentation process and the results obtained are included for several different cases. The benefits of parameterising the temperature profile using a few linear segments in order to obtain a smoother profile are considered with an additional smoothing procedure. Some other modifications to the original procedure for the optimisation from the previous chapter are also incorporated.

4.1. INTRODUCTION TO GENETIC ALGORITHMS

Genetic Algorithms (GAs) are search algorithms based on the mechanics of natural selection and natural genetics (Goldberg, 1989). GAs combine "survival of the fittest" among string structures with a structured, but still randomised, information exchange to form a search algorithm with some of the innovative style of human search. In every generation, a new set of artificial individuals (the strings) is created using bits and pieces of the fittest of them all; an occasional new part can be tried for good measure. They efficiently exploit historical information to speculate on new search points with expected improved performance.

Evolution strategies, evolutionary programming and genetic algorithms form the backbone of the field of evolutionary computation (Yao, 1999). In the 1950s and the 1960s, several computer scientists independently studied evolutionary systems with

the idea that evolution could be used as an optimisation tool for engineering problems. Later in the 1960s, evolution strategies were introduced. a method used to optimise real-valued parameter for devices such as airfoils (these idea was further developed in the 1970s). The field of evolution strategies has remained an active area of research, mostly developing independently from the field of genetic algorithms. Also in the 1960s, evolutionary programming was developed, a technique in which candidate solutions to given tasks were represented as finite-state machines, which were evolved by randomly mutating their state-transition diagrams and selecting the fittest.

With this, Genetic Algorithms were invented and developed at the University of Michigan in the 1960s and 1970s (Mitchell, 1998). In contrast with evolutionary strategies and evolutionary programming, the original goal was not to design algorithms to solve specific problems, but rather to formally study the phenomenon of adaptation as it occurs in nature and to develop ways in which the mechanisms of natural adaptation might be imported into computer systems.

Biologists have been intrigued with the mechanics of evolution since the evolutionary theory of biological change gained acceptance. Many people, biologists included, are astonished that life at the level of complexity that can be observed nowadays could have evolved in the relatively short time suggested by the fossil record. The mechanisms that drove this evolution are still not fully understood, but some of its features are known. Evolution takes place on chromosomes (organic devices for encoding the structure of living beings). A living being is created partly through a process of decoding chromosomes. The specifics of chromosomal encoding and decoding processes are not completely comprehended either but here are some general features of the theory that are generally accepted (Davis, 1991):

- *Evolution* is a process that operates on chromosomes rather than on the living beings they encode.
- *Natural selection* is the link between chromosomes and the performance of their decoded structures. Processes of natural selection cause those chromosomes that encode successful structures to reproduce more often than those that do not.

- The process of *reproduction* is the point at which evolution takes place. *Mutations* may cause the chromosomes of biological children to be different from those of their biological parents, and recombination processes may create quite different chromosomes in the children by combining material from the chromosomes of two parents.
- Biological evolution has no memory. Whatever it knows about producing individuals that will, function well in their environment is contained in the gene pool (the set of chromosomes carried by the current individuals) and in the structure of the chromosomes decoders.

Holland (1992) began to work on algorithms that manipulated strings of binary digits (1's and 0's) that he called chromosomes and carried out simulated evolution on populations of such chromosomes. Like in nature, his algorithms solved the problem of finding good chromosomes by manipulating the material in the chromosomes blindly and they knew nothing about the type of problem they were solving. The only information they were given was an evaluation of each chromosome they produced, and their only use of that evaluation was to bias the selection of chromosomes so that those with the best evaluations be likely to reproduce more often than those with bad assessment.

These algorithms, using simple encoding and reproduction mechanisms together with operators such as crossover, mutation and inversion, displayed complicated behaviour, and they turned out to solve some extremely difficult problems. In this case: *crossover* exchanges subparts of two chromosomes, roughly imitating biological recombination between two single-chromosome organisms; *mutation* randomly varies the allele values of some locations in the chromosome; and *inversion* reverses the order of a nearby section of the chromosome, as a result rearranging the order in which genes are displayed.

In the last several years there has been widespread interaction among researchers studying various evolutionary computation methods, and the boundaries between GAs, evolution strategies, evolutionary programming, and other evolutionary approaches have broken down to some extent. In order to unify concepts, Table 4.1 shows some frequently used terms in GAs:

Term	Representation
Chromosome	Vector which represents solutions of application task
Gene	Each solution which consists of a chromosome
Allele	Different possible settings for a trait
Selection	Choosing parents or offsprings chromosomes for the next generation
Individual	Each solution vector which is each chromosome
Population	Total individuals
Population size	The number of chromosomes
Fitness function	A function which evaluates how each solution is suitable to the given task
Diploid	Organisms whose chromosomes are arrayed in pairs
Haploid	Organisms whose chromosomes are unpaired
Recombination	Same as crossover
Gamete	A single chromosome
Phenotype	Expression type of solution values in task world
Genotype	Bit expression type of solution values used in GA

Table 4.1 Technical terms used in GA literatures

With this, there is not a unique accurate definition of "genetic algorithms" accepted by everyone in the evolutionary computation population that differentiates GAs from other evolutionary computation methods. Nevertheless, it can be said that most methods called GAs have at least the following elements in common: populations of chromosomes, selection according to fitness, crossover to produce new offspring, and random mutation of new offspring. Inversion is rarely used in today's implementations, and its advantages, if any, are not well established.

Some applications of genetic algorithms that come from variations of the basic scheme, have been used in a large number of scientific and engineering problems and models (Michalewicz, 1992).

The location of the Genetic Algorithms technique with respect to other deterministic and non-deterministic procedures is shown in the following hierarchy diagram. Figure 4.1 outlines the situation of natural techniques among other well known search procedures.

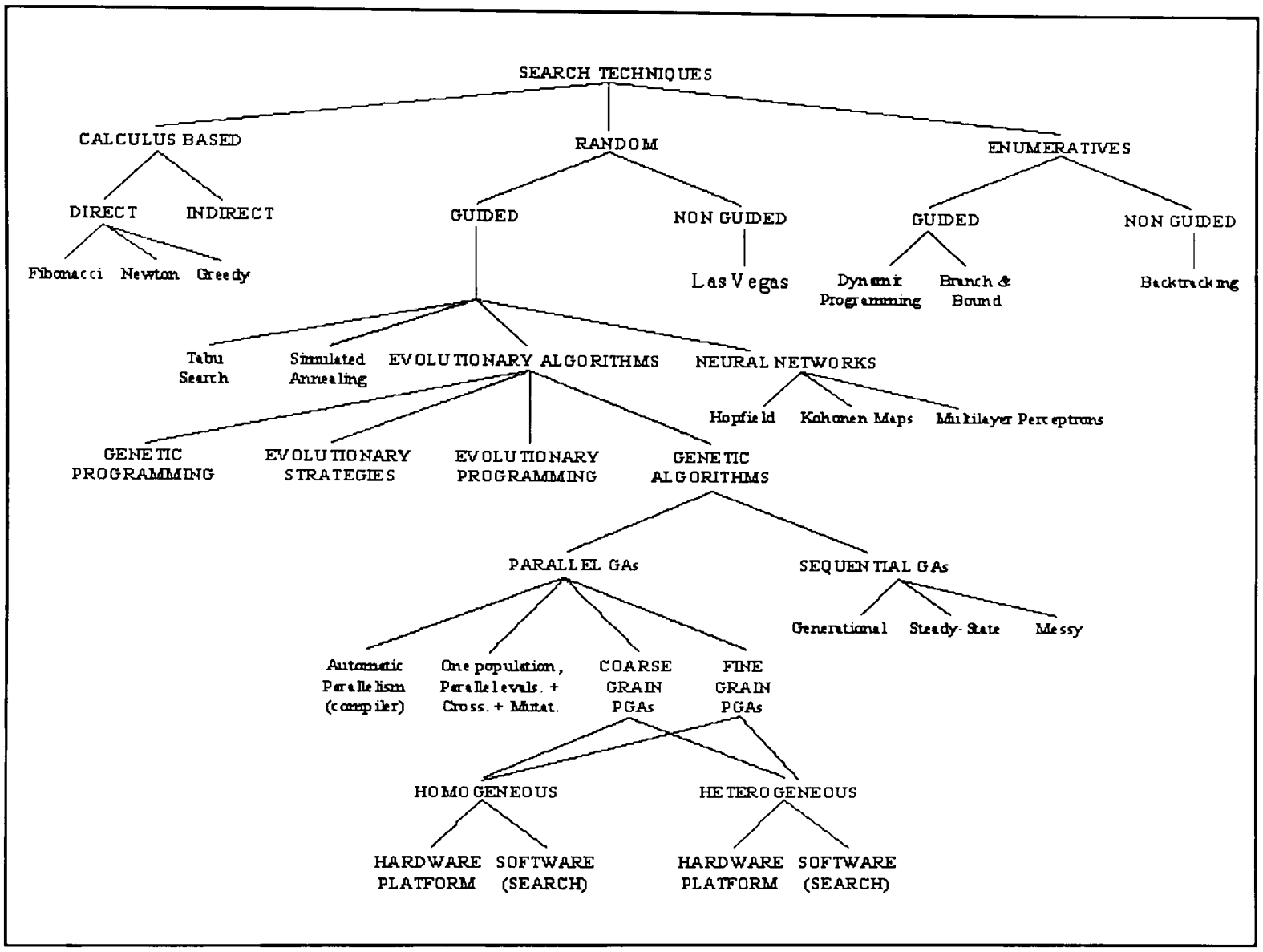

Figure 4.1 A possible classification of Search Techniques (Alba and Cotta, 1999).

4.2. OPTIMISATION USING GENETIC ALGORITHMS

In the case of linear programming, a global optimum can always be attained. Non-linear programming solvers generally use some form of gradient search technique to move along the steepest gradient until the highest point (maximisation) is reached (as seen in Chapter 3). However, these models may be subject to problems of convergence to local optima, or in some cases, may be unable to find a feasible solution (largely depends on the starting point of the solver). A starting point outside the feasible region may result in no feasible solution being found, even though feasible solutions may exist. Other starting points may lead to an optimal solution, but it is not possible to determine if it is a local or global optimum. Hence, the modeller can never be sure that the optimal solution produced using the model is the "true" optimum (Mardle and Pascoe, 1999).

Conventional search techniques are often incapable of optimising non-linear multimodal functions. In such cases, a random search method might be required. Ultimately, the search procedure finds a set of variables that optimises the fitness of an individual and/or of the whole population. With this, the GA technique has advantages over traditional non-linear solution techniques that cannot always achieve an optimal solution.

GA are most appropriate for complex non-linear models where location of the global optimum is a difficult task. It may be possible to use GA techniques to consider problems which may not be modelled as accurately using other approaches. Therefore, it appears to be a potentially useful approach. They do not use much knowledge about the problem to be optimised and do not deal directly with the parameters of the problem, GAs work with codes which represent parameters. The parameters to be optimised are usually represented in a string form since genetic operators are suitable for this type of representation (binary or integer representation method).

Some new modified models of genetic algorithms have been investigated recently (Pham and Karaboga, 2000); these include: Hybrid Genetic Algorithm, Cross

Breeding in Genetic Optimisation, GA with the Ability to Increase the Number of Alternative Solutions and GA with Variable Mutation Rates.

Additionally, in order to make the search operator more efficient, some modifications have been suggested (Davis, 1987):

- Two Crossover Points: The chromosome can be perceived as a circle with the first gene immediately obvious that there can be in fact two crossover points: one fixed at position zero and the other randomly selected. An immediate generalisation to the present crossover operator is to allow both crossover points to be randomly selected.

- Improved Crossover Implementation: The performance of a GA can be improved if the crossover is constrained to always produce variations whenever possible. This constraint can be implemented by simply restricting the location of crossover points. Crossover points are randomly selected for these reduced strings, and then mapped back into the original strings.

- Variable Crossover Rate: The idea is to go steady with crossover until variation has been absorbed, and then introduce more variation by increasing crossover again. A good measure for predicting allele loss is percent involvement. It is possible that by changing the crossover rate in response to changes in percent involvement, performance is improved.

When several criteria represented by the objective function are present simultaneously, the problem is said to be a Multi-Objective (MO) or multi-criteria optimisation problem (Goldberg, 1989). The majority of the engineering design problems are Multi-Objective, for the reason that there are several conflicting design aims which need to be simultaneously accomplished (Fonseca and Fleming, 1998 & Fleming and Chipperfield, 1998).

A Multiple-Objective optimum design problem is solved in a similar way to the Single-Objective (SO) problem . In a SO problem, the idea is to find a set of values for the design variables that yield (when subject to a number of constraints) an

optimum value of the objective/cost function. In MO problems, the designer tries to find the values for the design variables which optimise the objective functions at the same time, in this manner the solution is chosen from the so called Pareto optimal set; these solutions are generally in the boundary of the design region, or in the locus of the tangent points of the objective functions.

In general, for Multi-Objective problems the optimal solutions obtained by individual optimisation of the objectives (SO optimisation) are not a feasible solution to the Multi-Objective problem. Nonetheless, Multi-Objective optimisation as a computer aided design technique is able to watch over all the various conflicting design goals individually, but still compromising them simultaneously. Some computer tools (like the MOPS Toolbox for MATLAB Software) solve the problem by transforming the set of criteria into a weighted min-max optimisation problem, where the weights are chosen according to the demands (Joos, 1999).

4.3. GENETIC ALGORITHMS APPLIED TO THE BEER FERMENTATION PROCESS

In order to optimise the SIMULINK beer fermentation process modelled previously in Chapter 2, a new script file for MATLAB containing the necessary instructions from the Genetic Algorithm Toolbox (Chipperfield et al, 1994) have been created. In this *m-file* some initial parameters needed in the GAs toolbox have to be defined, i.e.: number of individuals per subpopulations, maximal number of generations, bounds on decision variables, crossover and mutation rate, etc. Some specific routines can also be chosen and/or changed here, such as: selection, recombination and mutation function for individuals. Is important to note that for this optimisation no initial profile had been used as a starting point. An extended description of the functions from this Genetic Algorithm Toolbox has been included in Appendix A.

The main objective function value (performance index) is obtained through the simulation itself by iterations between the SIMULINK model and the MATLAB script. This process can slow down the acquisition of the results of the optimisation, but it has to be done in this way to be compared with the previous results obtained with the Gradient Method and the DISOPE algorithm. It includes the following terms:

$$J_1 = +10 \cdot e_{end} \tag{2.24}$$

$$J_2 = -5.73 \times 10^{-8} \cdot e^{(95 \cdot diac_{end})} \tag{2.25}$$

$$J_3 = -1.16 \times 10^{-29} \cdot e^{(4.6 \cdot acet_{end})} \tag{2.26}$$

$$J_4 = -\int_0^{t_f} 9.91 \times 10^{-7} \cdot e^{(2.31 \cdot T)} dt \tag{2.27}$$

$$J_5 = -\sum_{i=1}^{160} \frac{|T_{i+1} - T_i|}{\Delta t} \tag{2.28}$$

These terms combined to obtain a cost function of the process, which is included in the SIMULINK model of Figure 4.2:

$$J = J_1 + J_2 + J_3 + J_4 + J_5 \tag{2.29}$$

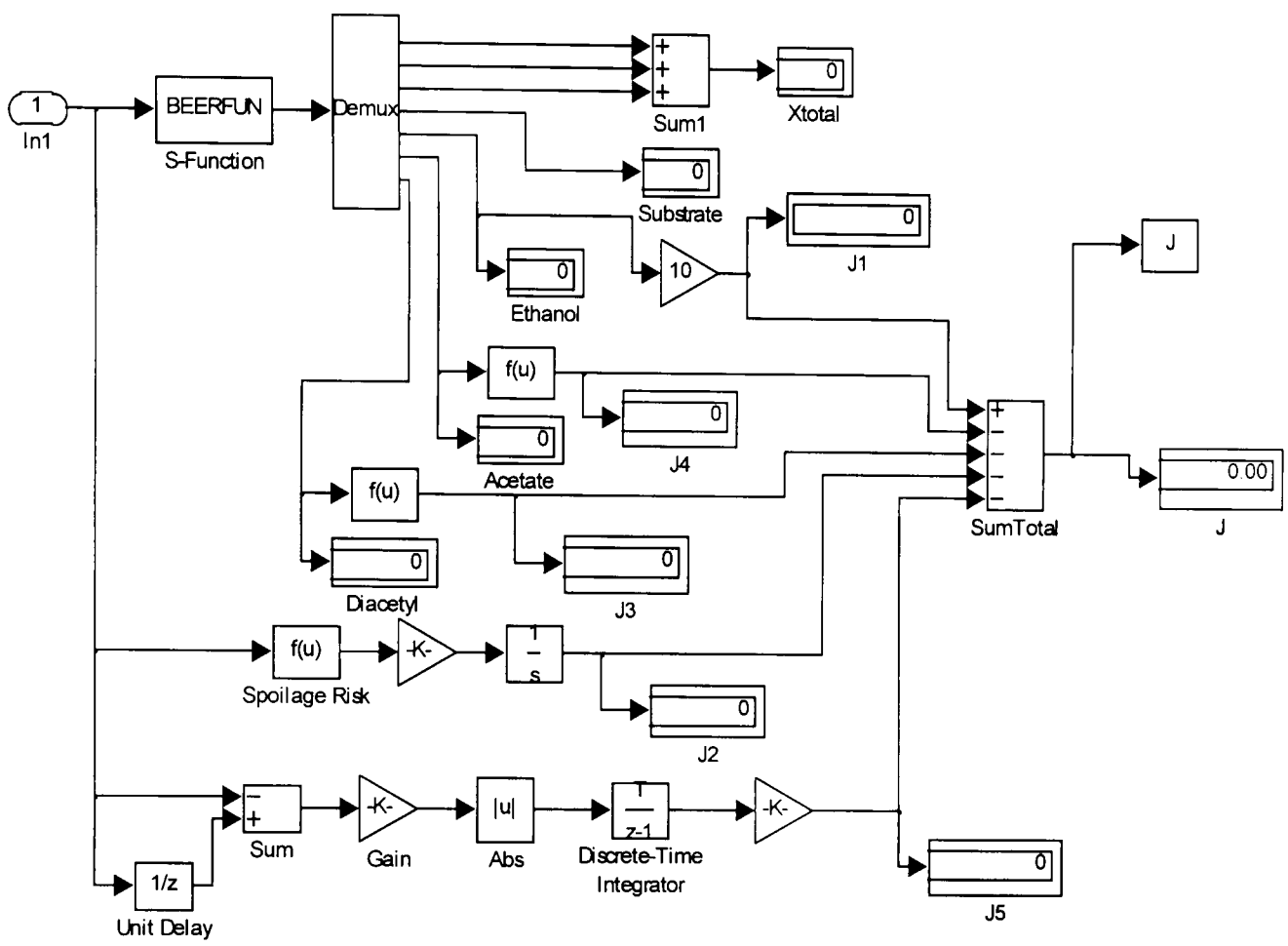

Figure 4.2 SIMULINK model "beernew" for the fermentation process

A reduced SIMULINK model named "beer" has also been created, this model does not include the last term of the cost function (J_5); this have been done trying to accelerate the optimisation with GAs and also see the effect on the temperature profile obtained.

First tests have been completed with a low number of individuals and generations. The bounds of the decision variables have been set from 0 to 16°C according to previous experiences in the industry. The aim is to achieve a good value of the objective function and a good/smooth temperature profile, without incurring in too much time involved for the optimisation.

No initial temperature profile has been used for the optimisation and different input parameter values have been used to see the effect that any change may produce. With this in mind, several runs have been made and good results have been observed in obtaining the input temperature profile desire for industrial application.

The termination criteria for the optimisation of the simulated beer model using the GA Toolbox has been set by the total amount of generations required for every case considered. Thus, the total time required for every optimisation will depend basically on the number of individuals and generations allowed. This total time is considered to be an important factor in order to be used as a measure of comparison with the other techniques/methods used. With this, although GAs benefits are known to involve more computational effort than other techniques; it has been chosen appropriate due to the off-line nature of the optimisation required for the beer fermentation process.

Every case considered using the same initial parameters has been optimised not one but three times; this has been done with the aim of selecting the best candidate temperature profile that provides the maximum objective function value from these three runs. Herewith, results have changed slightly (as expected) from one run to the other in the order of approximately 2-3% but nonetheless, the more significant improvements have been observed when these initial parameters are altered (different cases).

These results are included with some of the initial parameter values used in Table 4.2; in that order: Case number, number of Individuals, number of Generations, Generation Gap, SIMULINK Model used for the optimisation, Maximum performance index *J* value obtained and Total ime required.

The final plots resulting from the application of the new input profile obtained (Figures 4.3 to 4.14). Some functions and parameters remained unchanged for all cases: The *Selection* function used was roulette wheel selection, the *Recombination* function chosen was double point crossover, the *Mutation* rate was set by one divided by the number of decision variables (1/NVAR), the *Crossover* rate was fixed

to 0.8 and the Minimum and Maximum boundaries were set to 0 and 16°C respectively.

Case	Individual	Generation	Generation Gap	Model used	Max. *J* value	Total time (hours)*
1	200	200	0.7	beer	579.38	3.872
2	300	200	0.8	beer	583.14	5.784
3	300	250	0.7	beer	582.20	7.112
4	500	250	0.8	beer	587.59	11.855
5	600	200	0.8	beer	596.18	11.826
6	750	250	0.8	beer	599.47	18.538
7	1000	300	0.8	beer	601.06	29.731
8	600	200	0.7	beernew	397.85	12.145
9	800	250	0.7	beernew	460.62	21.375
10	1000	300	0.7	beernew	457.28	29.411
11	1500	350	0.8	beernew	528.99	53.590
12	2000	400	0.8	beernew	541.77	89.324

*Using a 800MHz CPU with 256MB RAM

Table 4.2 Results obtained with GA for Cases 1 to 12

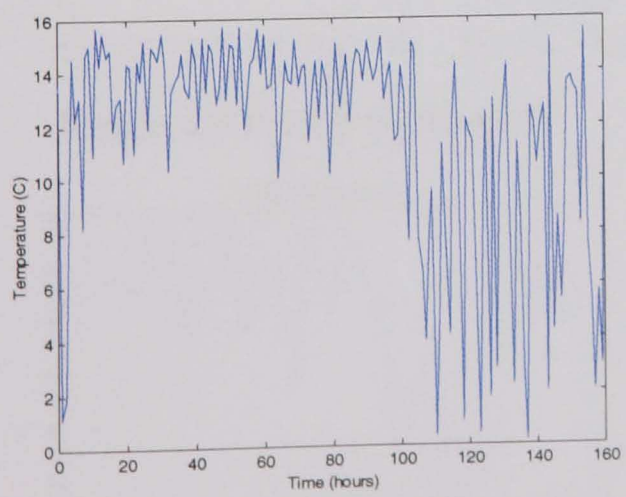

Figure 4.3 Profile for Case #1

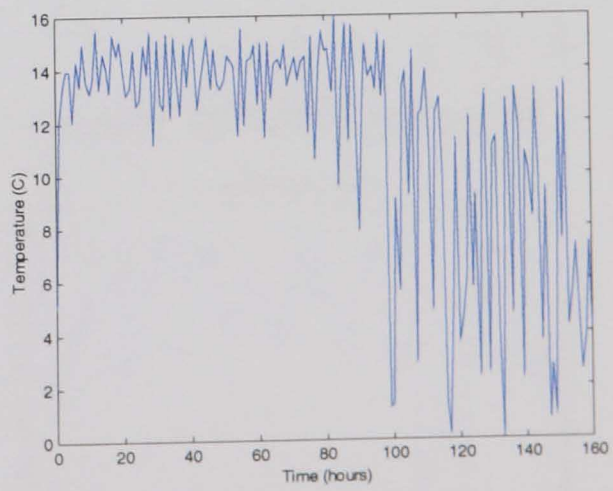

Figure 4.4 Profile for Case #2

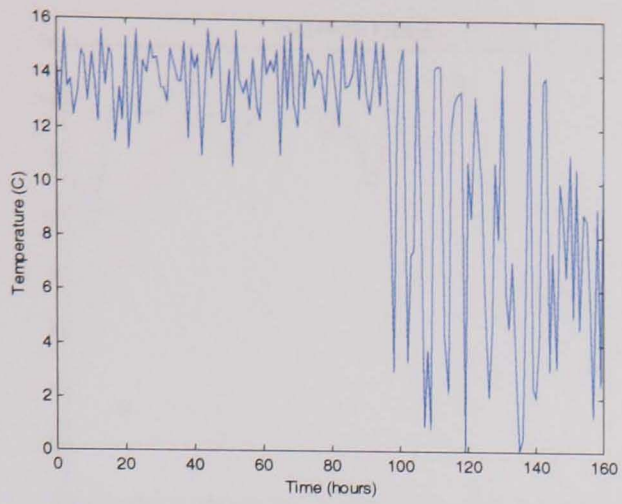

Figure 4.5 Profile for Case #3

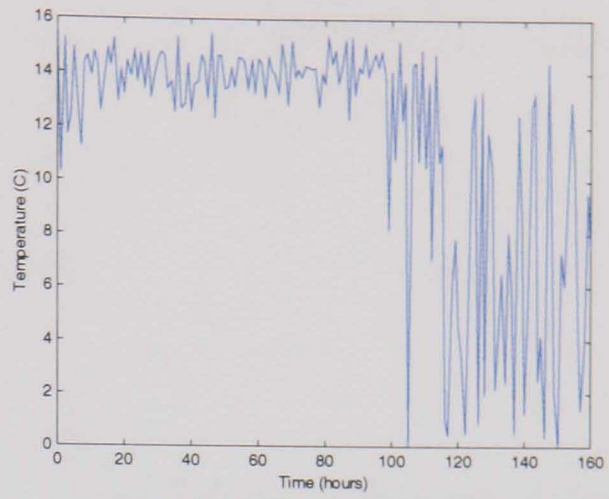

Figure 4.6 Profile for Case #4

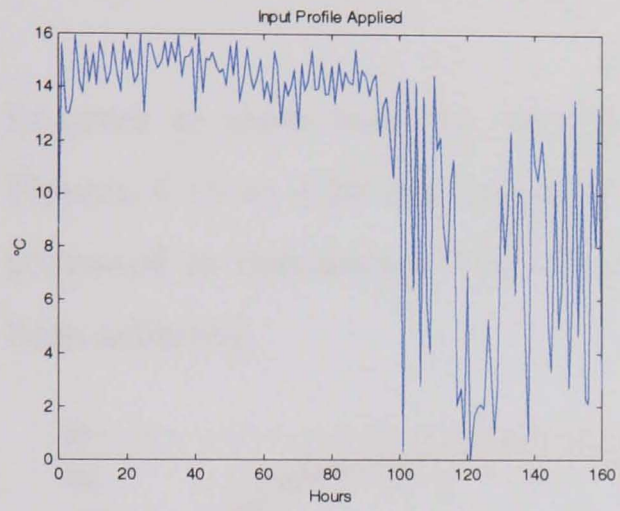

Figure 4.7 Profile for Case #5

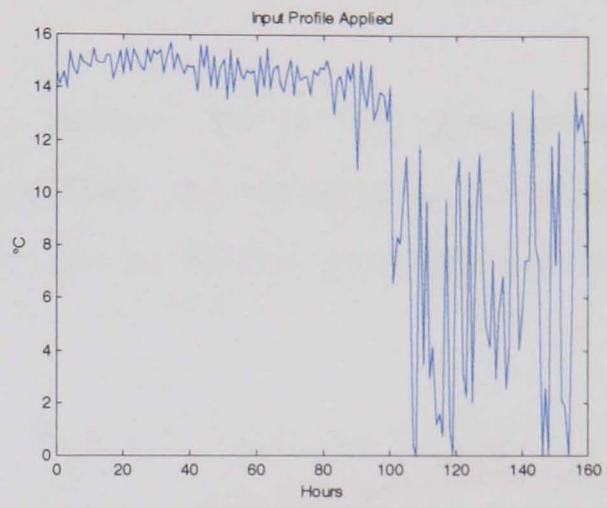

Figure 4.8 Profile for Case #6

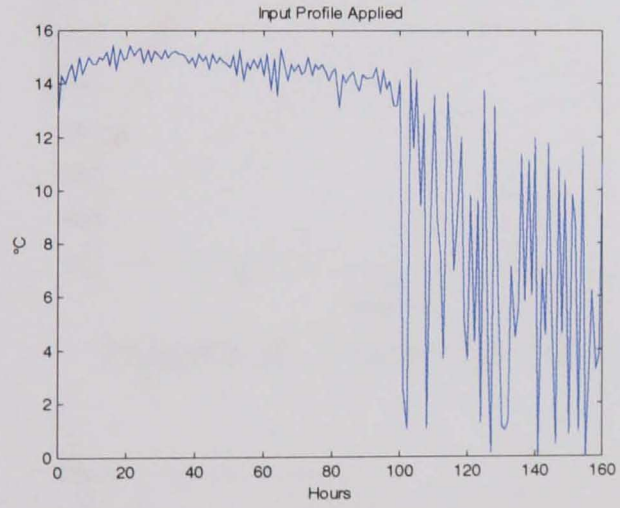

Figure 4.9 Profile for Case #7

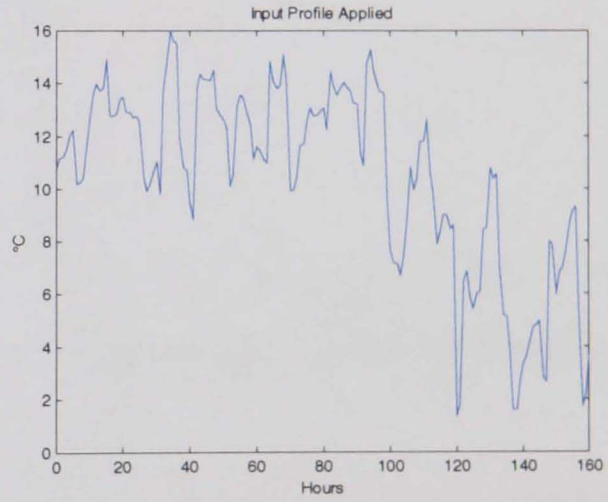

Figure 4.10 Profile for Case #8

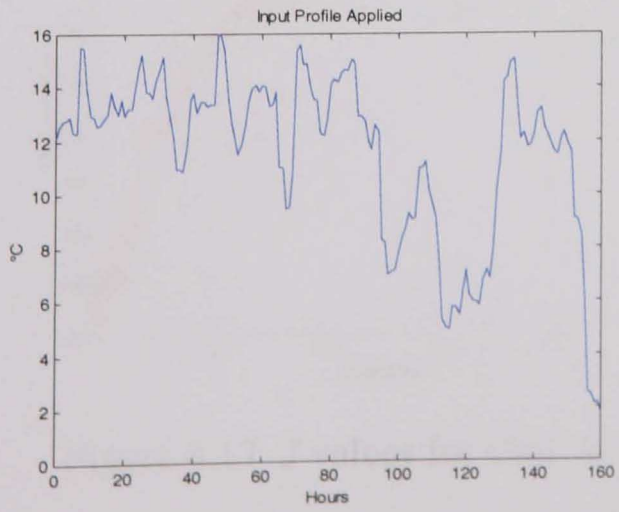

Figure 4.11 Profile for Case #9

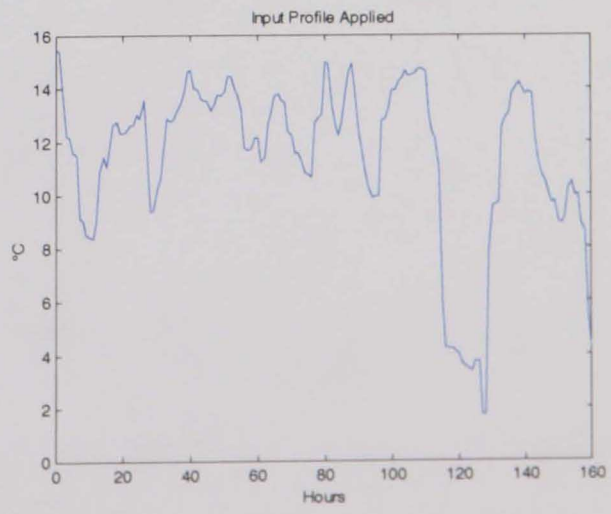

Figure 4.12 Profile for Case #10

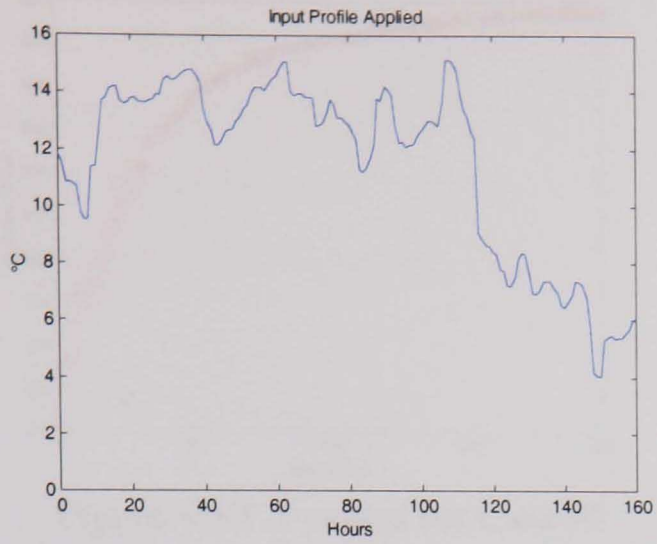

Figure 4.13 Profile for Case #11

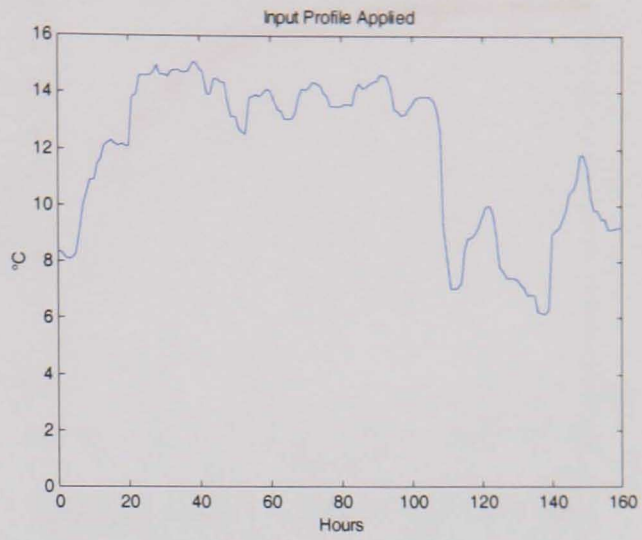

Figure 4.14 Profile for Case #12

In order to show how the optimal profile is attained and the GA development, Figures 4.15 to 4.26 are included. For this purpose, the twelve cases have been presented to demonstrate how the maximum value of the Objective Function has been achieved.

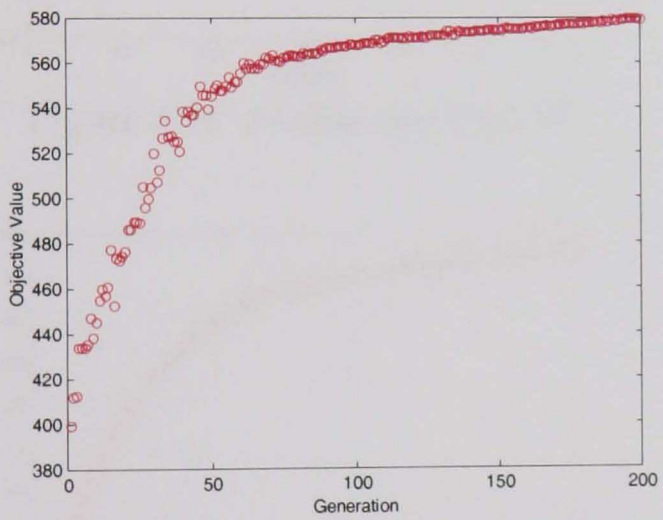

Figure 4.15 *J* values for Case #1

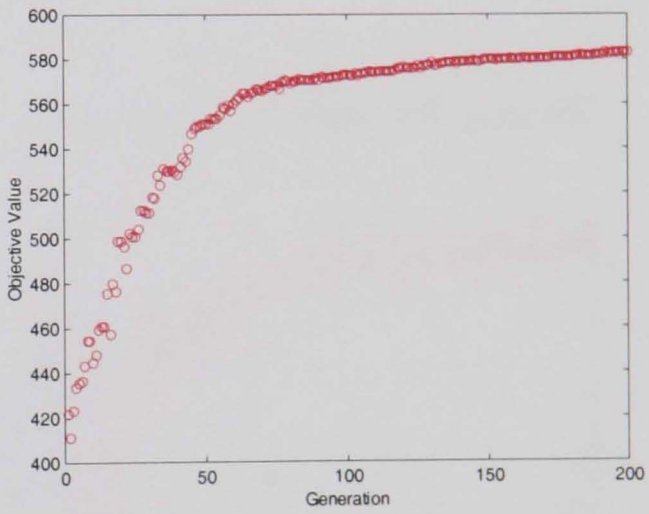

Figure 4.16 *J* values for Case #2

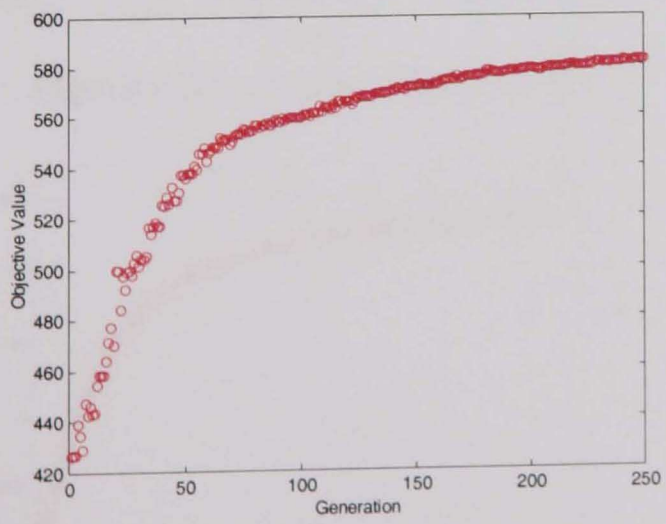

Figure 4.17 *J* values for Case #3

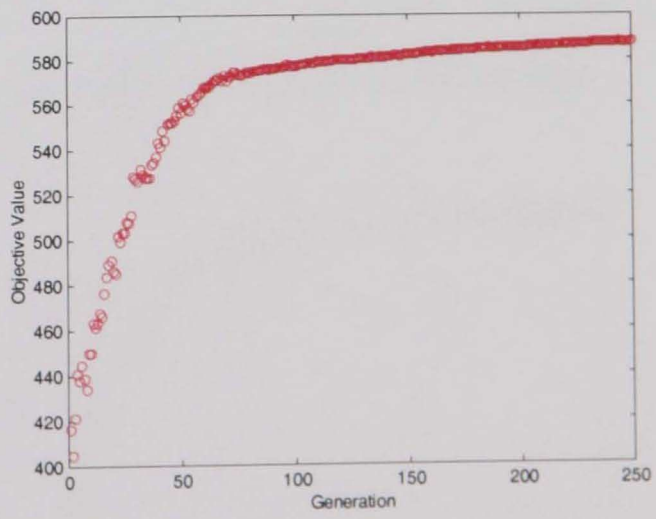

Figure 4.18 *J* values for Case #4

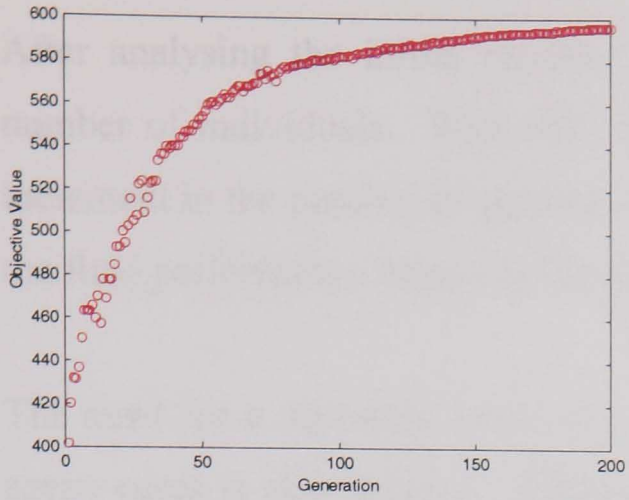

Figure 4.19 *J* values for Case #5

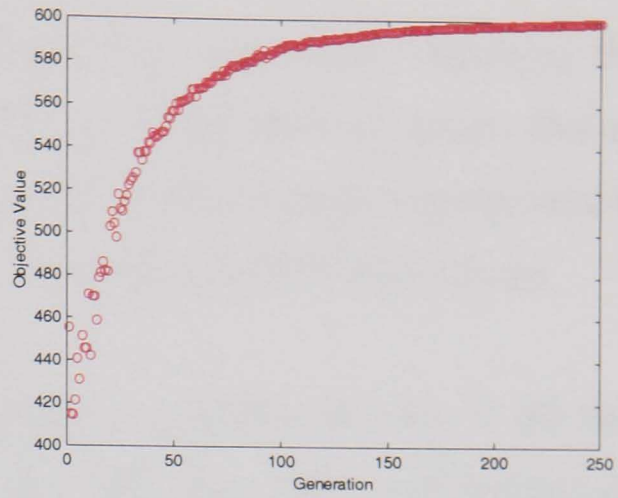

Figure 4.20 *J* values for Case #6

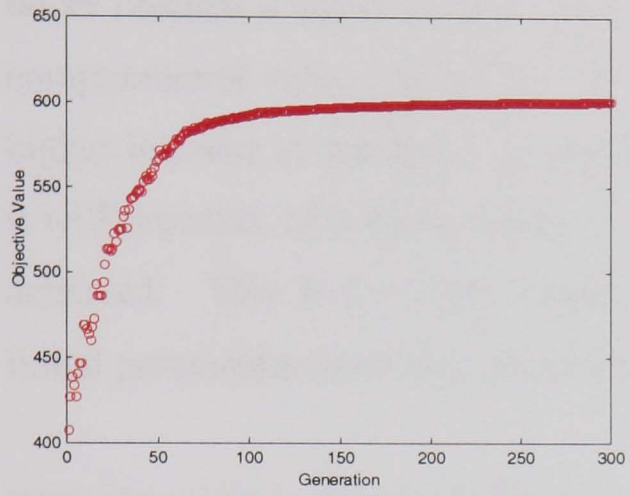

Figure 4.21 *J* values for Case #7

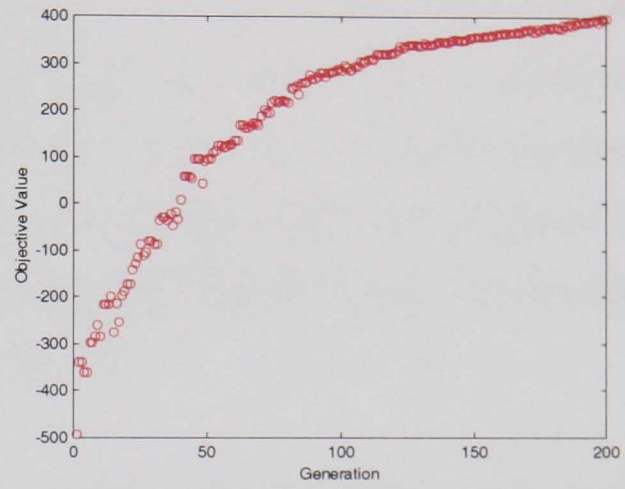

Figure 4.22 *J* values for Case #8

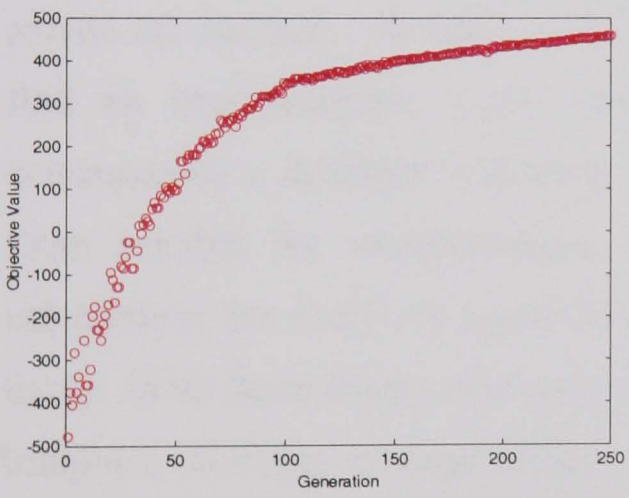

Figure 4.23 *J* values for Case #9

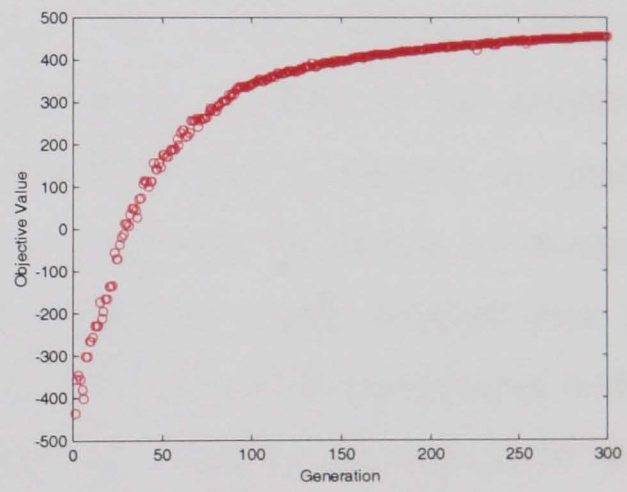

Figure 4.24 *J* values for Case #10

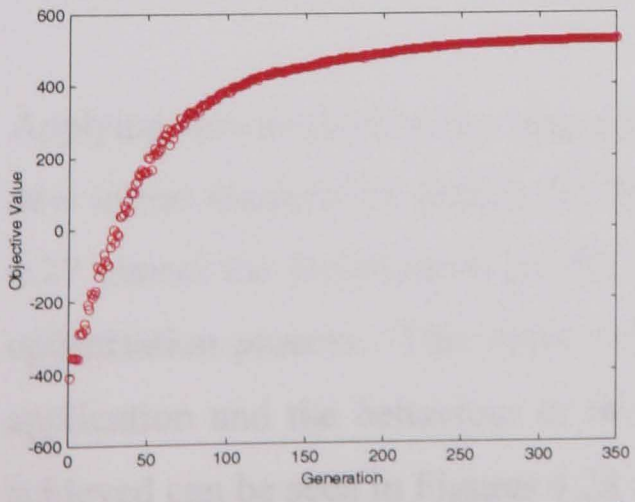

Figure 4.25 *J* values for Case #12

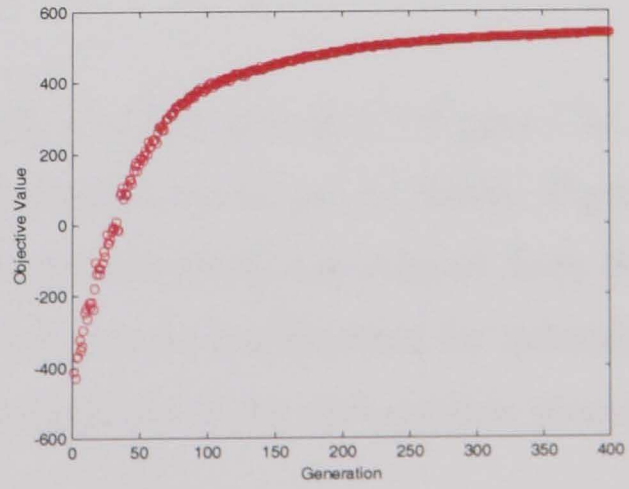

Figure 4.26 *J* values for Case #12

After analysing the initial results, further analysis has been made increasing the number of individuals. With this, it can be seen from the previous figures that an increment in the number of generations is not going to make a major improvement in the final performance value and can incur in a very long optimisation procedure.

The need for a smoother solution in the optimised temperature profiles of the first seven cases is also evident. After applying once more the penalty sub-function J_s, for Cases #8 to #12, the new temperature profiles obtained show better silhouette but never reached a superior final cost function value. It is important to note that the computational time needed for the optimisation of these cases was considerably higher in order to achieve a good objective function value (in the order of days) and it will probably take weeks before a similar result (in value) to the one in Case #7 is achieved. This is the main reason why the optimisations with new and increase initial parameters have been concluded after Case #12.

With the results obtained after the application of GAs for the optimisation of the fermentation process it can be seen that with more time spent in the simulation better results are reached. However, a large amount of time will be required in order to find an implementable input temperature profile to be used in the industry, consequently a different smoothing process should be used to improve and make them suitable for implementation. This has been done by means of average calculations for every 40 hours of the initial temperature profile obtained (four in total). With these results, four new temperature values are obtained and placed in the midpoint of every average range. Horizontal lines from the previous point and straight lines to the next one have been linked to obtain a piece-wise linearisation. Another MATLAB script has been developed to achieve this purpose.

Applying this method to the temperature profile obtained in Case #7 (Figure 4.9), a new initial temperature profile for the beer simulation process can be found. Figure 4.27 shows the temperature profile obtained after smoothing the original from the optimisation process. This improved profile can now be implemented for industrial application and the behaviour of the model parameters of the fermentation process achieved can be seen in Figures 4.28 to 4.31.

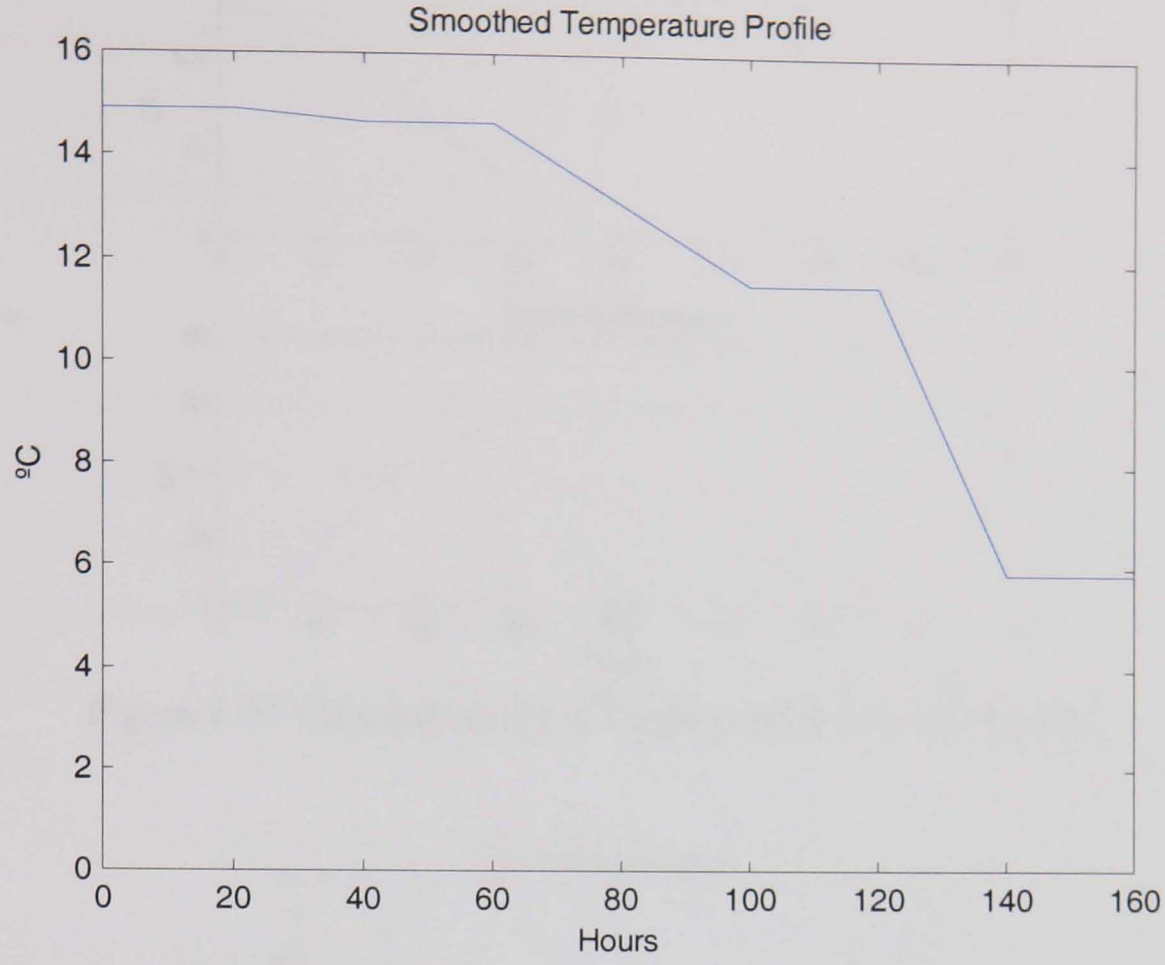

Figure 4.27 Smoothed Temperature (°C) vs. Time (Hours)

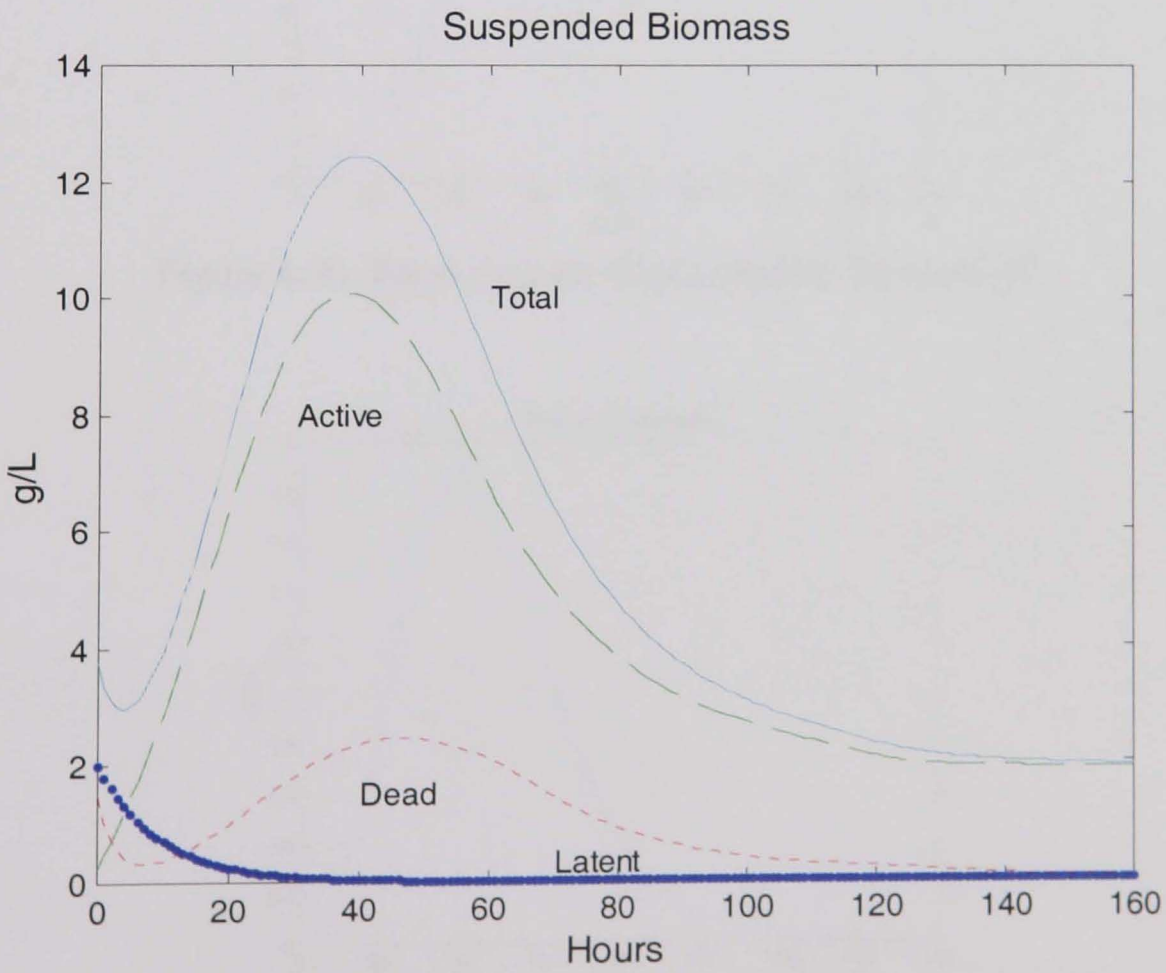

Figure 4.28 Suspended Biomass Behaviour for Case #7

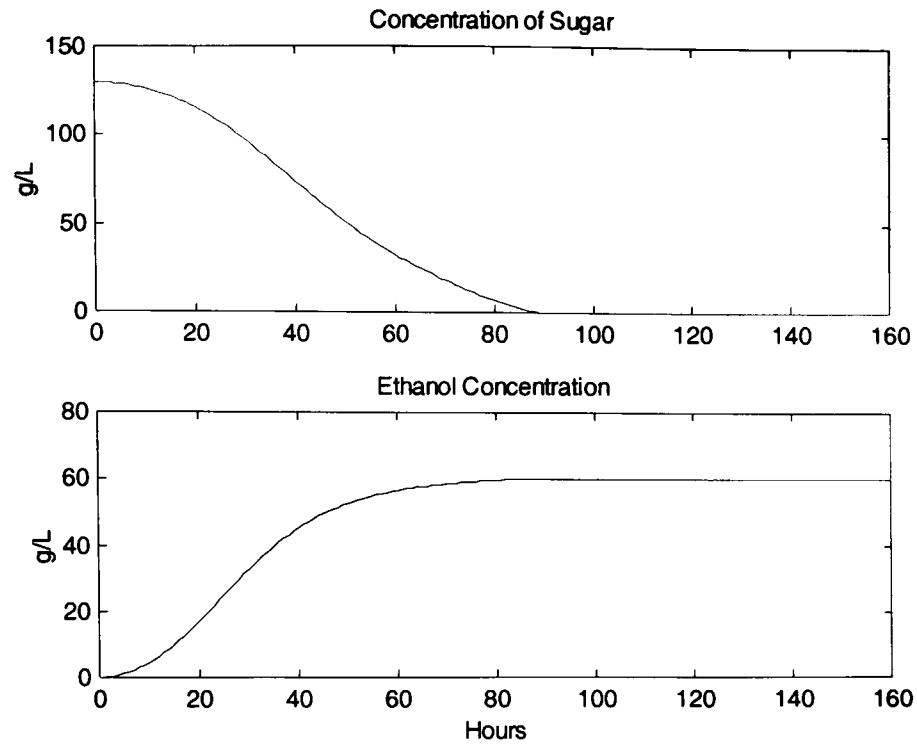

Figure 4.29 Concentrations of Sugar and Ethanol for Case #7

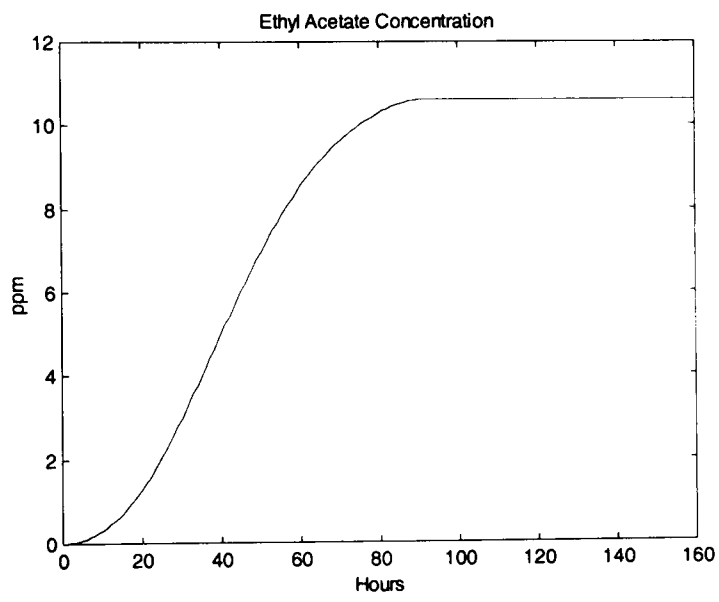

Figure 4.30 Ethyl Acetate Concentration for Case #7

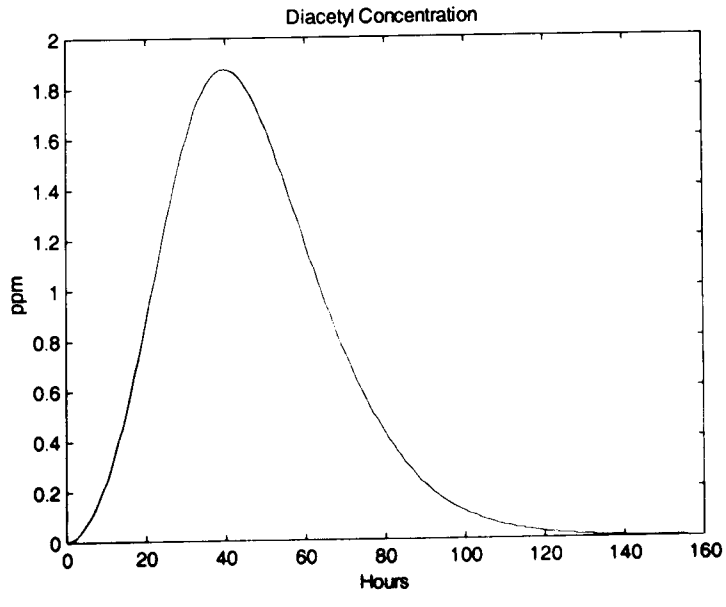

Figure 4.31 Diacetyl Concentration for Case #7

With this new temperature profile the maximum objective function value obtained for Case #7 (601.06) has decreased to a new value of 589.7745: but still gives and increment of nearly 14% if compared with the original value obtained in Chapter 2 for the industrial temperature profile (518.90).

However, even with these results a new approach to obtain a smoother temperature profile in less time without requiring an extra procedure after the optimisation is pursued. This has been achieved by means of changing the step size used for the fermentation time; this is, instead of one step/evaluation per hour, the total fermentation time is divided between the number of Decision Variables chosen (i.e. 10 decision variables in a 160 hour fermentation provides one step/evaluation every 16 hours). Table 4.3 summarises the scenarios for the cases considered. Once again the functions and parameters not included remain unchanged as in Cases 1 to 12.

Case	Individuals	Generations	Generation Gap	Decision Variables	Maximum J value	Total time (minutes)*
A	100	50	0.6	5	596.94	28.721
B	100	50	0.6	8	592.83	29.008
C	100	50	0.6	10	588.12	28.930
D	200	50	0.7	5	597.63	59.342
E	200	50	0.7	8	596.03	59.843
F	200	50	0.7	10	596.58	59.135
G	300	100	0.8	5	600.09	178.777
H	300	100	0.8	8	600.11	176.142
I	300	100	0.8	10	596.77	175.933
J	500	150	0.8	5	600.13	444.007
K	500	150	0.8	8	599.99	443.180
L	500	150	0.8	10	599.18	440.389

*Using an 800MHz CPU with 256MB RAM

Table 4.3 New results obtained with GA for Cases A to L

In order to assess the new obtained temperature profiles with the development of the performance index, Figures 4.32 to 4.55 show the Objective Value (J) beside the Optimised Temperature Profile attained for the best chromosome found for each case.

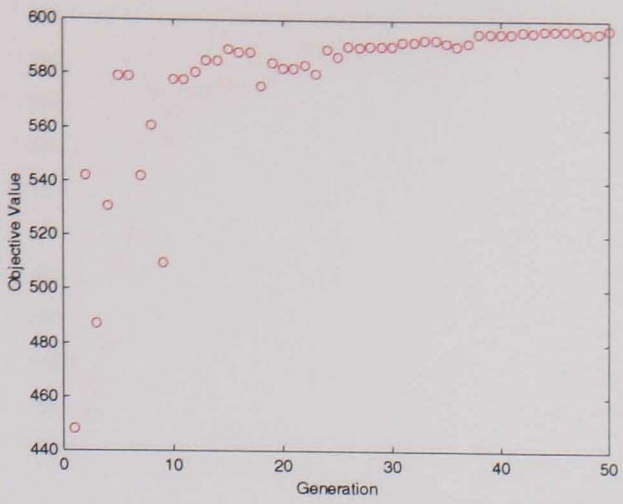

Figure 4.32 *J* values for Case A

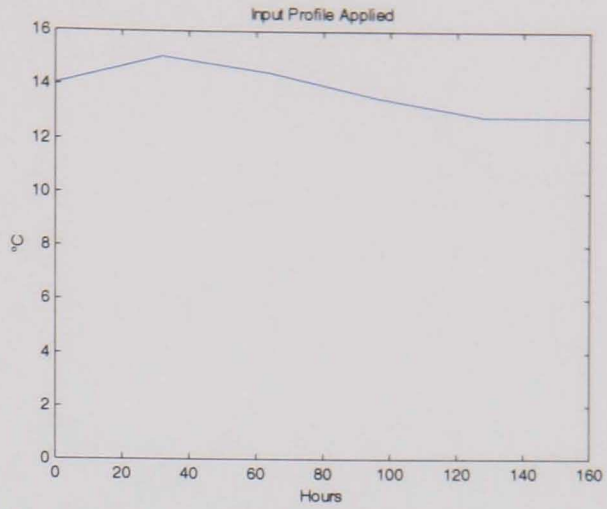

Figure 4.33 Profile for Case A

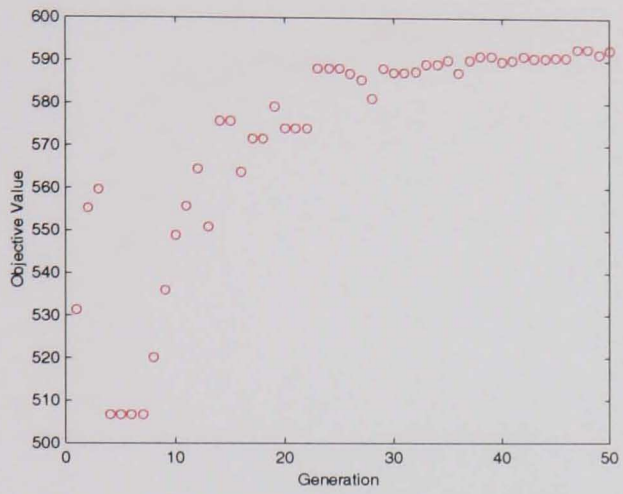

Figure 4.34 *J* values for Case B

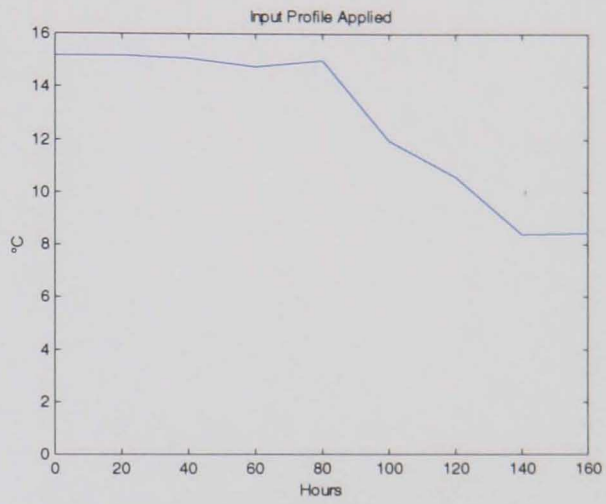

Figure 4.35 Profile for Case B

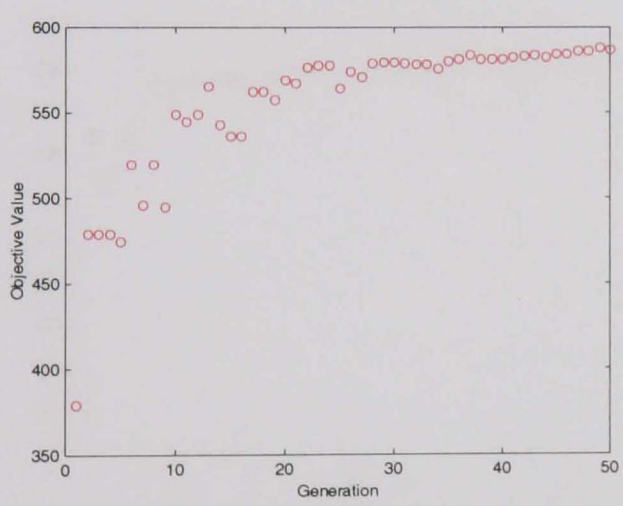

Figure 4.36 *J* values for Case C

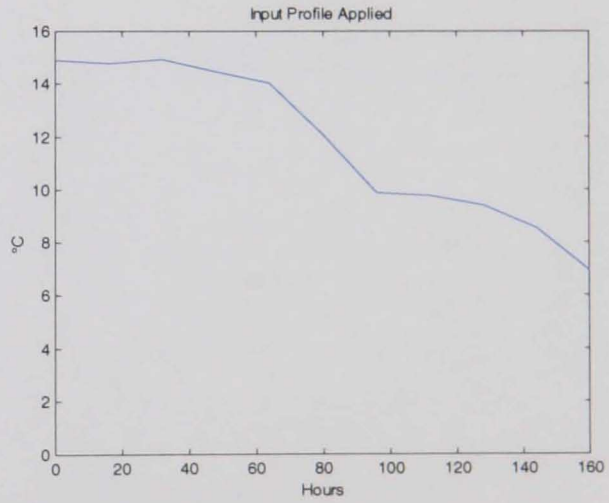

Figure 4.37 Profile for Case C

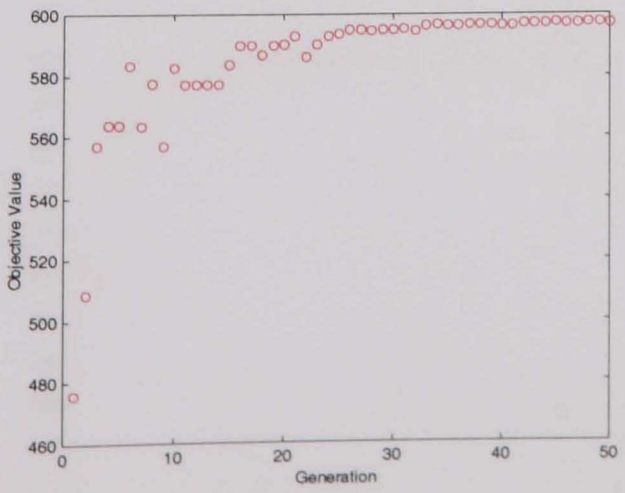

Figure 4.38 *J* values for Case D

Figure 4.39 Profile for Case D

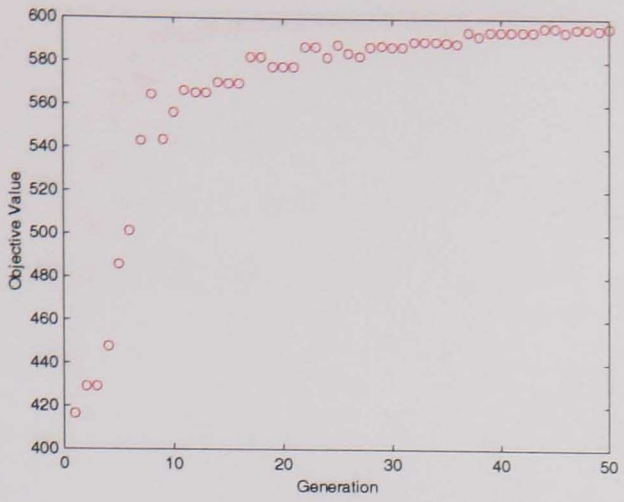

Figure 4.40 *J* values for Case E

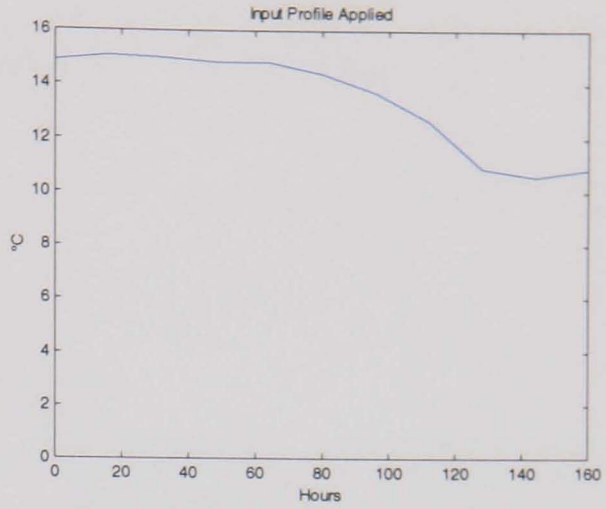

Figure 4.41 Profile for Case E

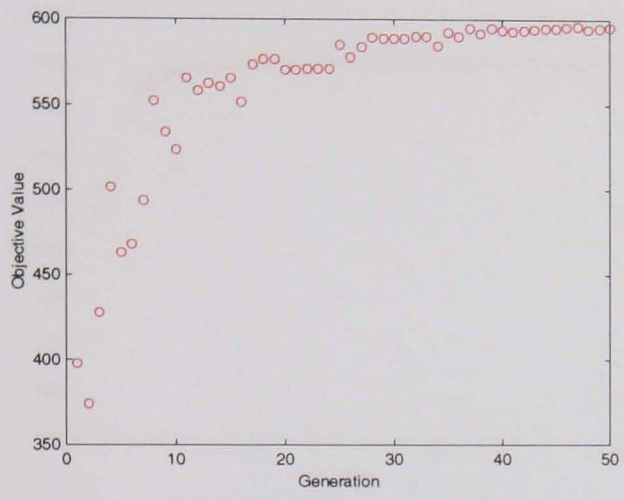

Figure 4.42 *J* values for Case F

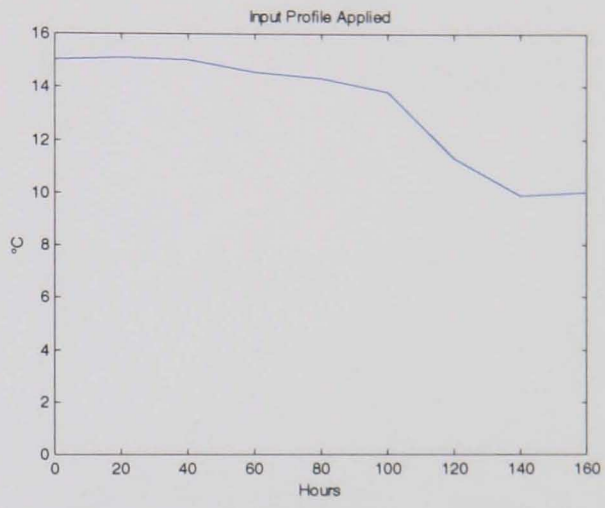

Figure 4.43 Profile for Case F

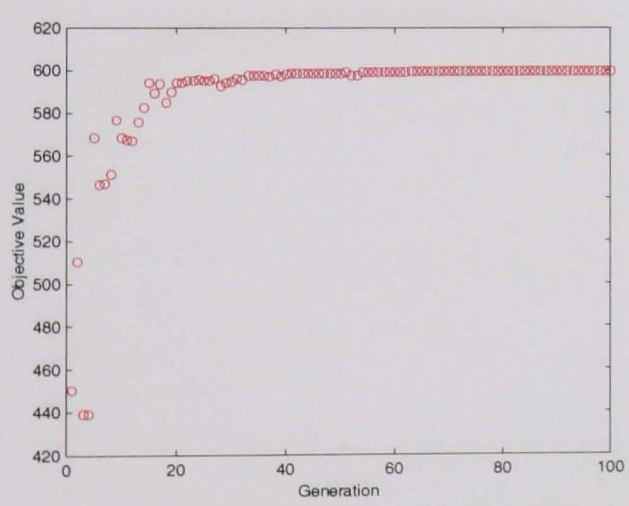

Figure 4.44 *J* values for Case G

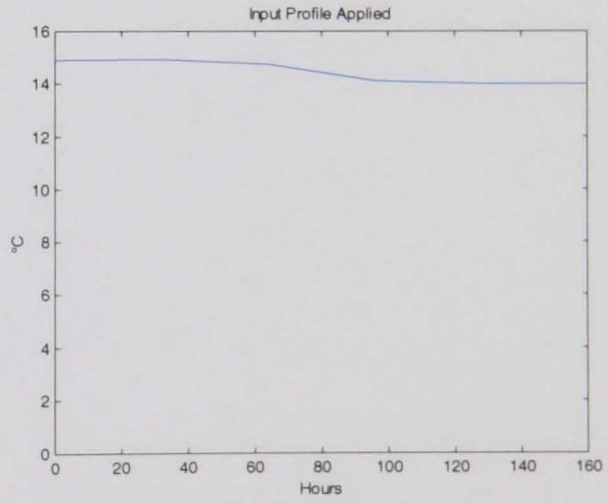

Figure 4.45 Profile for Case G

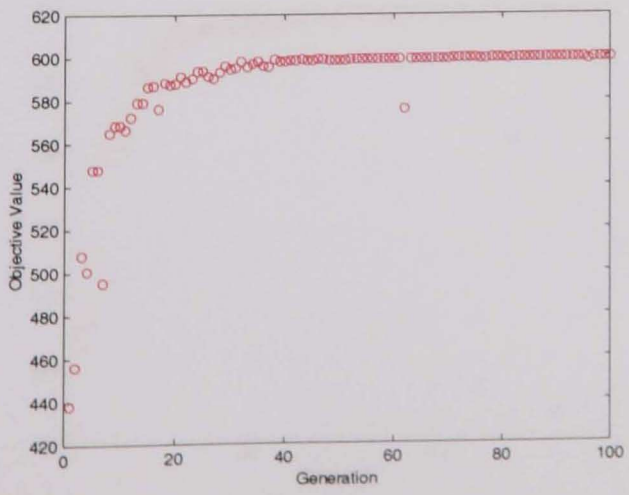

Figure 4.46 *J* values for Case H

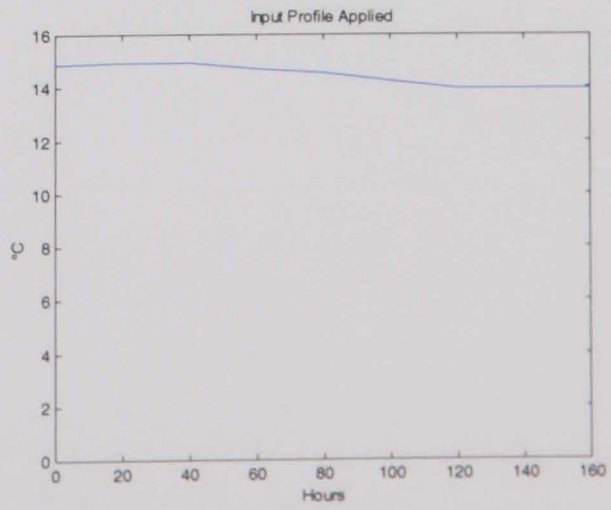

Figure 4.47 Profile for Case H

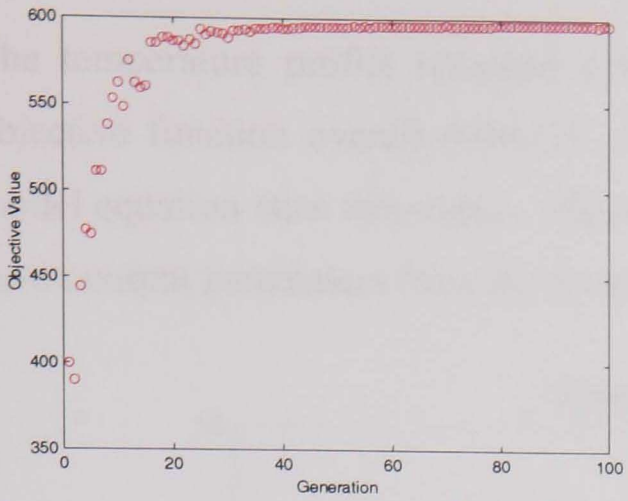

Figure 4.48 *J* values for Case I

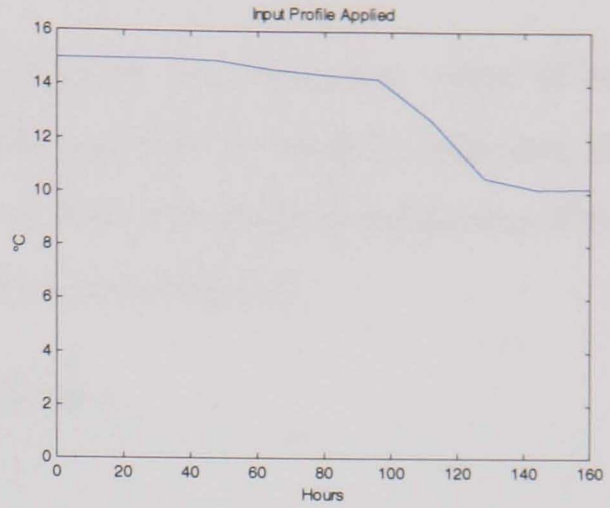

Figure 4.49 Profile for Case I

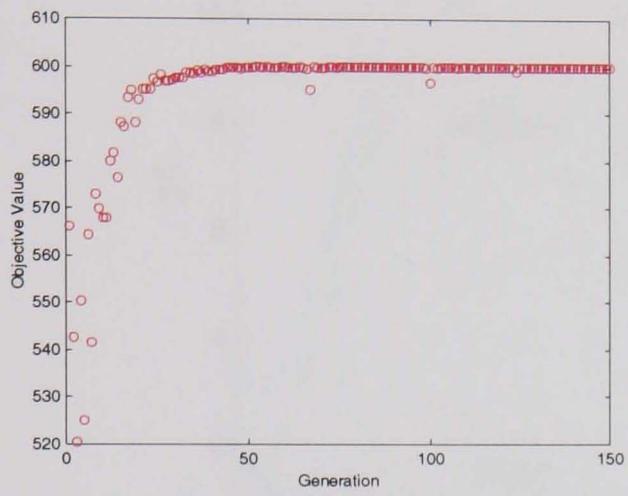

Figure 4.50 *J* values for Case J

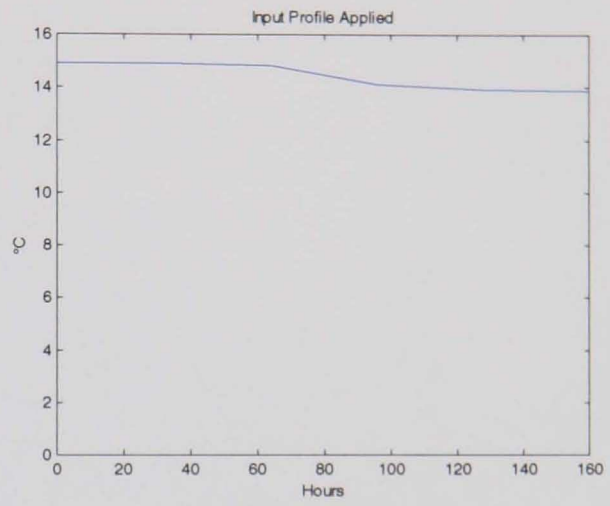

Figure 4.51 Profile for Case J

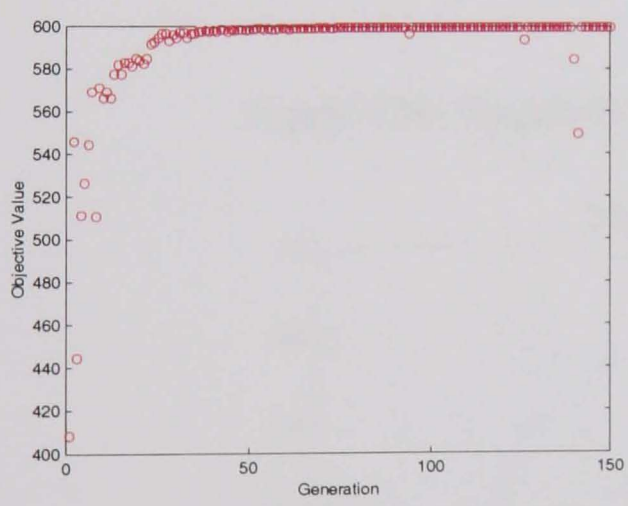

Figure 4.52 *J* values for Case K

Figure 4.53 Profile for Case K

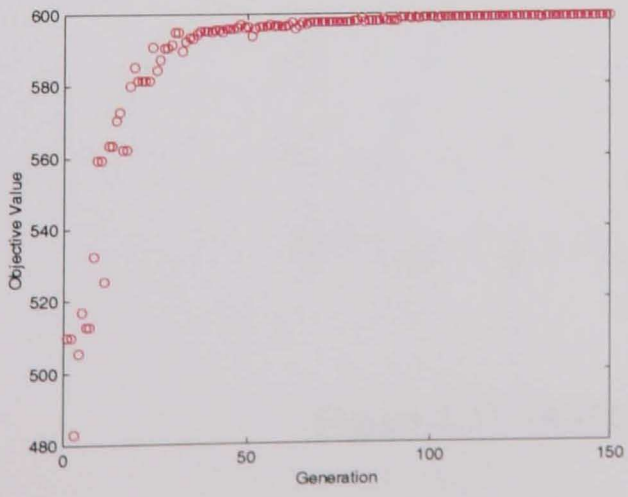

Figure 4.54 *J* values for Case L

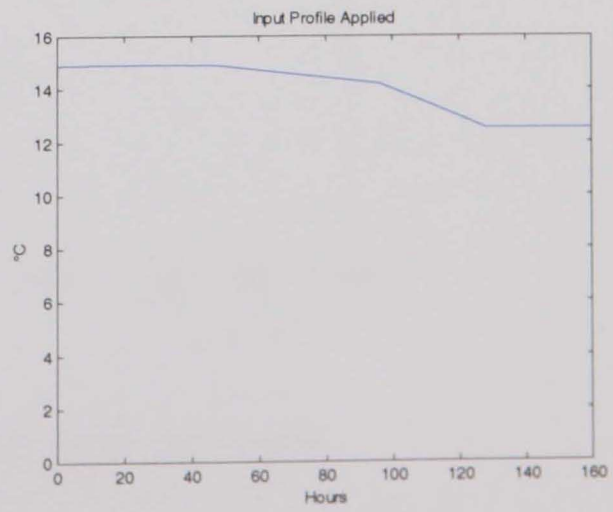

Figure 4.55 Profile for Case L

The temperature profile obtained from Case J gives the maximum value of the objective function overall (600.13); this profile can then be used to represent the model equation state responses. Figures 4.56 to 4.60 represent the behaviour of the seven output parameters from the fermentation process modelled.

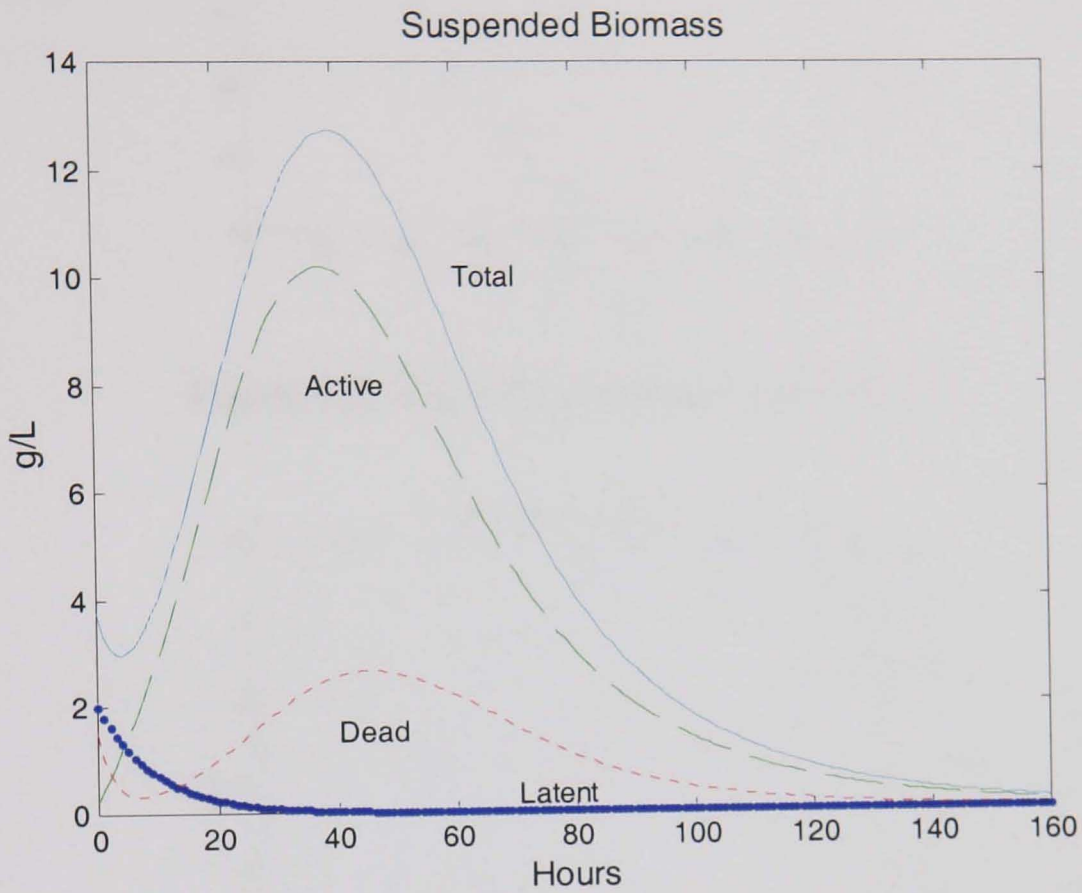

Figure 4.56 Suspended Biomass Behaviour for Case J

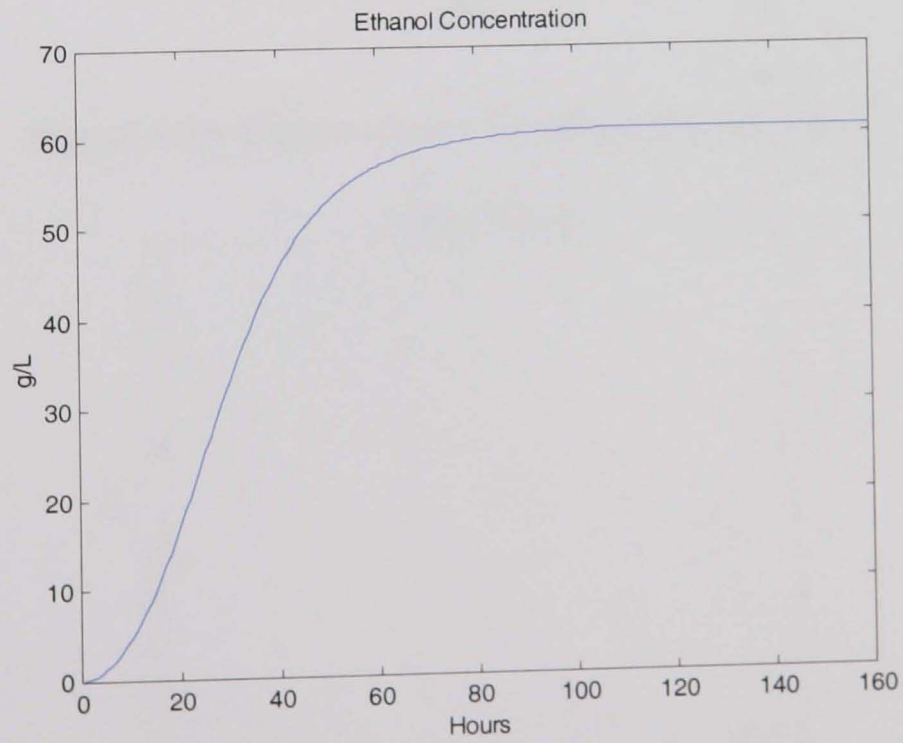

Figure 4.57 Ethanol Concentration for Case J

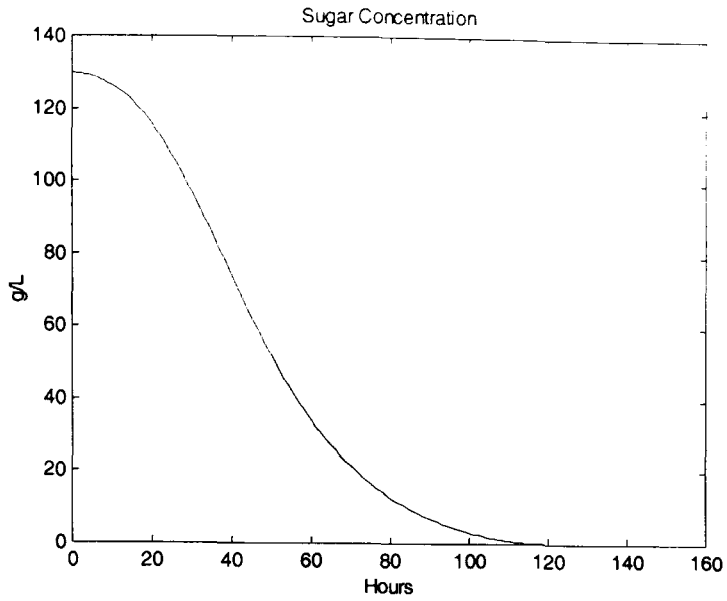

Figure 4.58 Sugar Concentration for Case J

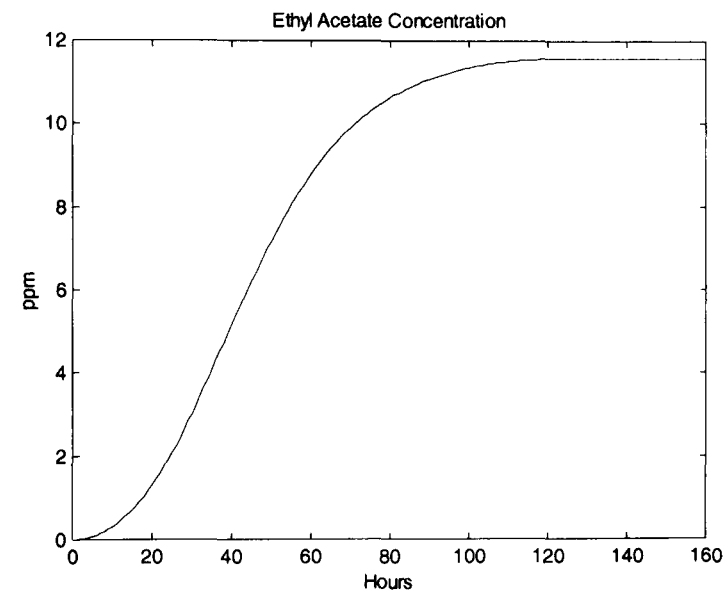

Figure 4.59 Ethyl Acetate Concentration for Case J

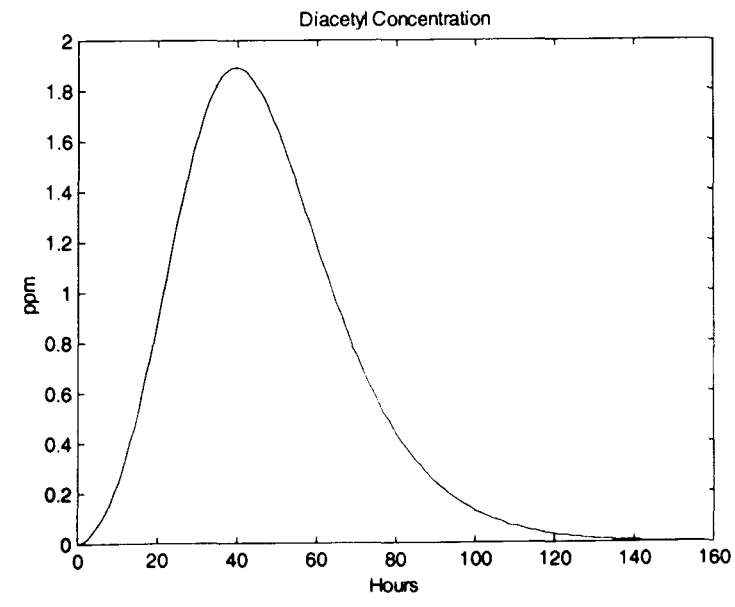

Figure 4.60 Diacetyl Concentration for Case J

It can clearly be seen in this optimised temperature profile that no smoothing technique is required. This profile does not includes abrupt changes so that the industry can implement it in practice. This has been achieved thanks to just five decision variables (one step/evaluation every 32 hours) that produce small changes in the temperature profile. Thus, this makes the sub-function J_5 small enough not to be so influential in the overall performance index value.

4.4 SUMMARY

Genetic Algorithms have proved to be suitable in the optimisation of fermentation processes and no previous knowledge, such an initial temperature profile, has been necessary to obtain a satisfactory result.

Previous work by Andres-Toro et al (1997b) in order to achieve a better profile for implementation have been acceptable and encouraging. The SIMULINK implementation described in this Chapter appears to be a useful and accurate representation of the model that was easy to interface with the Genetic Algorithm Toolbox by Chipperfield et al (1994). In addition, a better objective function value has been obtained with the Genetic Algorithm Toolbox for the optimisation of the beer process. Also a softer profile by parameterising and calculating average temperatures made results suitable for implementation in practice.

CHAPTER V

SEQUENTIAL QUADRATIC PROGRAMMING (SQP)

Optimisation by means of a Sequential Quadratic Programming routine is the main objective of this chapter. Constrained Optimisation of non-linear processes is introduced as part of understanding the principles surrounding this technique. MATLAB's Optimization Toolbox (Coleman et al, 1999) is explained briefly and in particular how it deals with SQP. SQP is then applied to the fermentation process selected (with some changes to the original optimisation problem) and the results obtained are also presented.

5.1 NON-LINEAR PROGRAMMING WITH CONSTRAINTS

Non-linear optimisation problems have applications in chemical engineering, trajectory optimisation, mechanical structure design and many other areas. Optimisation problems with non-linear constraints are significantly more difficult to solve than unconstrained or linearly constrained optimisation problems. For instance, algorithms to solve inequality constrained non-linear problems may take many iterations and function evaluations, with each function evaluation being an expensive operation in some cases. Performance in practice can be measured by counting iterations, function evaluations, gradient evaluations and the computational cost of determining an iterate.

The general aim in constrained optimisation is to convert the problem into an easier sub problem that can then be solved and used as the basis of an iterative process. A characteristic of a large class of early methods is the translation of the constrained problem to a basic unconstrained problem by using a penalty function for constraints, which are near or beyond the constraint boundary (Coleman et al, 1999). A wide variety of methods have been proposed for solving constrained optimisation problems; these include reduced-gradient methods, penalty and barrier function

methods, methods based on augmented Lagrangians or projected augmented Lagrangians and SQP methods (Goldsmith, 1999). Some of these methods form quadratic approximations to composite functions of the objective function and constraints, this being the Lagrangian in the case of SQP methods.

Quadratic Programming (QP) is the name given to the procedure that minimises a quadratic function of n variables subject to m linear inequality or equality, or both types, of constraints. A quadratic programming problem is the simplest form of non-linear programming problem with inequality constraints (Edgar and Himmelblau, 2001). Each inequality constraint must either be satisfied as an equality or it is not involved in the solution of the QP problem, so that once the binding constraints are identified, the QP technique can reduce to a vertex-searching procedure examining the intersection of linear equations as in linear programming.

A general Nonlinear Optimisation Problem (NLP) with inequality and equality constraints can be written as follows:

$$\begin{array}{lll} \text{minimise} & f(x) & \\ \text{subject to} & g_i(x) \le 0, & i \in I = \{1,\cdots, m\} \\ & h_j(x) = 0, & j \in J = \{1,\cdots, p\} \\ & x \in C & \end{array} \qquad (5.1)$$

where $x \in \Re^n$, $C \subseteq \Re^n$ is a certain set and f, $g_1, \cdots, g_m, h_1, \cdots, h_p$ are functions defined on C (or on an open set that contains the set C). I and J are index sets. The set of feasible solutions will be denoted by $\Im$, consequently:

$$\Im = \{ x \in C \mid g_i(x) \le 0, i = 1,\cdots, m \text{ and } h_j(x) = 0, j = 1,\cdots, p \}$$

There is a large literature on necessary conditions for constrained optimisation problems, given in terms of various generalized derivatives, usually in the presence of additional constraint qualifications (Borwein et al, 1996; Coleman et al, 1999 and Ross et al, 1997).

Methods for solving constrained nonlinear problems are based on seeking a point satisfying conditions that hold at a local minimiser. Essential to the understanding of these conditions is the Lagrangian L, a function in the variables x and the Lagrange multipliers λ, defined as:

$$L(x,\mu,\lambda) = f(x) + \sum_{j=1}^{p} \mu_j h_j(x) + \sum_{i=1}^{m} \lambda_i g_i(x)$$
$$= f(x) + \mu^T h(x) + \lambda^T g(x) \qquad (5.2)$$

The Lagrangian is used to express first-order and second-order optimality conditions for a local minimiser. The first-order optimality conditions are also known as the Karush-Kuhn-Tucker (KKT) conditions. The KKT theorem states that under certain constraint qualification conditions, the solution x^* of the NLP is a stationary point of the NLP (known as a KKT point). The KKT point of the NLP is the point x for which there exist $\lambda \in \Re^m$, $\mu \in \Re^p$ such that:

(i) $\nabla L_x(x,\mu,\lambda) = 0$

(ii) $h(x) = 0$

(iii) $g(x^*) \leq 0$

(iv) $(\lambda)^T g(x^*) = 0$

In general, the solution of the KKT equations forms the basis to many non-linear programming algorithms that attempt to compute directly the Lagrange multipliers. Constrained Quasi-Newton methods guarantee super-linear convergence by accumulating second order information regarding the KKT equations using a quasi-Newton updating procedure. These methods are commonly referred to as Sequential Quadratic Programming (SQP) methods since a QP sub problem is solved at each major iteration. SQP is also known as Iterative Quadratic Programming, Recursive Quadratic Programming, and Constrained Variable Metric Methods.

Sequential Quadratic Programming is one of the most effective methods for nonlinearly constrained optimisation; it can be used both in line search and trust-region frameworks, and it is appropriate for small or large problems. Unlike

sequential linearly constrained methods, which are effective when most of the constraints are linear, SQP methods show their strength when solving problems with significant non-linearities.

Most SQP methods are normally infeasible methods, meaning that neither the initial point nor any of the subsequent iterates need to be feasible (Nocedal and Wright, 1999). This can be advantageous when the problem contains significantly non-linear constraints, because it can be computationally expensive to stay inside the feasible region. In some applications, however, the functions that make up the problem are not always defined outside of the feasible region.

5.2 SEQUENTIAL QUADRATIC PROGRAMMING

The following general constrained non-linear programming problem is now considered:

$$\begin{array}{lll} \text{minimise} & f(x) & x = [x_1 \quad x_2 \quad \cdots \quad x_n]^T \\ \text{subject to} & h_j(x) = 0 & j = 1, \cdots, p \\ & g_i(x) \le 0 & i = 1, \cdots, m \end{array} \qquad (5.3)$$

The Lagrange function (from Equation 5.2) includes the objective function and equality and inequality constraints with the additional multipliers:

$$L(x, \mu, \lambda) = f(x) + \sum_{j=1}^{p} \mu_j h_j(x) + \sum_{i=1}^{m} \lambda_i g_i(x)$$

An optimal solution to the general problem must satisfy the Karush-Kuhn-Tucker (KKT) conditions for optimality described in the previous section. These conditions serve as the basis for the design of some algorithms and as termination criteria for others (Edgar and Himmelblau, 2001). Table 5.1 shows these optimality conditions.

The *necessary conditions* for x^* to be a local minimum of $f(x)$ are:
a) $f(x)$, $h_j(x)$, $g_i(x)$ are all twice differentiable at x^*
b) The so-called "second-order constraint qualification" holds; the sufficient conditions for this requirement are that the gradients of the binding constraints ($g_i(x^*) = 0$), $\nabla g_i(x^*)$, and the equality constraints, $\nabla h_j(x^*)$, because $h_j(x^*) = 0$, are linearly independent.
c) The Lagrange multipliers exist; they do if b) holds
d) The constraints are satisfied at x^*
1. $h_j(x^*) = 0$
2. $g_i(x^*) \le 0$
e) The Lagrange multipliers λ_i^* (at x^*) for the inequality constraints are not negative (μ_j can be positive or negative)
$\lambda_i^* \ge 0$
f) The binding (active) inequality constraints are zero; the inactive inequality constraints are > 0.

and the associated λ_i's are 0 at x^*

$$\lambda_i^* g_i(x^*) = 0$$

g) The Lagrangian function is at a stationary point

$$\nabla L_x(x^*, \mu^*, \lambda^*) = 0$$

h) The Hessian matrix of L is positive semi-definite for those $\mathbf{v}$'s which $\mathbf{v}^T \nabla g_i(x^*) = 0$, and

i) $\mathbf{v}^T \nabla h_j(x^*) = 0$, that is, for all the active constraints

$$\mathbf{v}^T \nabla^2 [L(x^*, \mu^*, \lambda^*)]\mathbf{v} \geq 0$$

The *sufficient conditions* for x^* to be a local minimum are:

j) The necessary conditions a), b) by implication, c), d), e), f), and g)

k) Plus a modification of necessary condition h):

The Hessian matrix of L is positive definite for these vectors $\mathbf{v}$ such that

$$\left.\begin{aligned} \mathbf{v}^T \nabla g_i(x^*) &= 0 \\ \mathbf{v}^T \nabla h_j(x^*) &= 0 \end{aligned}\right\} \quad \text{for the binding constraints}$$

$$\mathbf{v}^T \nabla g_i(x^*) \geq 0 \qquad \text{for the inactive constraints}$$

$$\mathbf{v}^T \nabla^2 [L(x^*, \mu^*, \lambda^*)]\mathbf{v} > 0$$

Table 5.1 The necessary and sufficient conditions for x^* to be a local minimum of the general constrained non-linear programming problem.

If the Newton's method for solving equations is applied to satisfy the Karush-Kuhn-Tucker necessary conditions for a non-linear programming problem containing only equality constraints, the Lagrange function is:

$$L(x, \mu) = f(x) + \sum_j \mu_j h_j(x) \tag{5.4}$$

and accordingly, the Karush-Kuhn-Tucker necessary conditions are:

$$\nabla L = \nabla f(x) + \sum_j \mu_j \nabla h_j(x) = 0 \tag{5.5}$$

$$h_j(x) = 0$$

Newton's method applied to the above two equations (Eqs. 5.4 and 5.5) produces:

$$\begin{bmatrix} \nabla^2 L & -J \\ -J^T & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \mu \end{bmatrix} = \begin{bmatrix} -\nabla L \\ h \end{bmatrix} \tag{5.6}$$

In this case, J stands for the Jacobian matrix of the equality constraints. This system of linear equations is solved for Δx and $\Delta \mu$. Powell (1978) showed that if Δx and $\Delta \mu$ satisfy the two linear equations, then they satisfy the necessary optimality of the following quadratic-programming problem to determine the search direction $\boldsymbol{s}$:

$$\begin{aligned} &\text{minimise} && F(s) = s^T \nabla f(x) + \frac{1}{2} s^T B s && \\ &\text{subject to} && g_i(x) + s^T \nabla g_i(x) \le 0 && i = 1,\ldots,m \\ & && h_j(x) + s^T \nabla h_j(x) = 0 && j = 1,\ldots,p \end{aligned} \tag{5.7}$$

where B is a positive definite approximation of the Hessian matrix of the Lagrange function $L(x)$. The constraints are linearised constraints and the objective function is quadratic. If B is used instead of $\nabla^2 f(x)$, super-linear convergence can be achieved.

By including the inequality constraints in the Lagrange function these restrictions can be treated. However, the Hessian Matrix of the Lagrange function may not be positive definite; in this case B is used instead via a suitable updating formula such as the BFGS formula (developed by Broyden, Fletcher, Goldfarb and Shanno; and included in Powell, 1978) that requires only the first derivatives so that second derivatives do not have to be computed.

In relation to the SQP implementation, two ways of applying the SQP method for solving the general non-linear programming problem can be taken into consideration (Nocedal and Wright, 1999). The first approach solves at every iteration the quadratic subprogram, taking the active set at the solution of this sub problem as a guess of the optimal active set. This approach is referred to as the IQP (Inequality constrained Quadratic Programming) approach, and has proved to be quite successful in practice. Its main disadvantage is the cost of solving the general quadratic program, which can be high when the problem is large. As the iterates of the SQP method converge to the solution, however, solving the quadratic sub problem

becomes very economical if information is been carried from the previous iteration to make a good guess of the optimal solution of the current sub problem. This is the case considered in Equation 5.7.

The second approach selects a subset of constraints at each iteration to be the so called working set, and solves only equality-constrained sub problems where the constraints in the working sets are imposed as equalities and all other constraints are ignored. The working set is updated at every iteration by rules based on the Lagrange multiplier estimates, or by solving an auxiliary sub problem. This EQP (Equality constrained Quadratic Programming) approach has the benefit that the equality-constrained quadratic sub problems are easier to solve and require less sophisticated software:

$$\begin{aligned} &\text{minimise} && s^T \nabla f(x) + \frac{1}{2} s^T B s \\ &\text{subject to} && g_j(x) + s^T \nabla g_j(x) = 0 \\ & && h_j(x) + s^T \nabla h_j(x) = 0 \qquad j = 1,\ldots,q \end{aligned} \tag{5.8}$$

where q is the number of active inequality plus equality constraints (linearised at $x^{(k)}$).

The main difference in the two procedures is that in the EQP solution the set of active constraints is unknown so that an active set strategy must be determined by an additional procedure. Usually, the active set of the IQP predicts correctly the active set of the non-linear programming problem itself in the vicinity of x^* for any bounded positive definite B. Also, the Lagrange multipliers move towards the multipliers of the non-linear programming problem as $x \to x^*$.

A relationship exists between the EQP sub problem to get s and a penalty function P. The essential idea of this penalty function P is to transform Equation 5.3 into a problem where a single unconstrained function is minimised (Edgar and Himmelblau, 2001):

$$\left.\begin{array}{ll} \text{minimise} & f(x) \\ \text{subject to} & h(x) = 0 \\ & g(x) \le 0 \end{array}\right\} \Rightarrow \text{minimise } P(f.\, h,\, g\,)$$

For the particular case of the EQP considered, this penalty function can be redefined including a scalar weight ***r*** as the penalty factor. As long as the value of the penalty is zero at the solution x^*, the solution is $P(x^*) = f(x^*)$ as wanted. Hence Equation 5.9 can be defined as:

$$P(x,r) = f(x) + \frac{1}{r}\hat{h}^T(x)\hat{h}(x) \qquad (5.9)$$

where $\hat{h}(x)$ designates the vector of all the equality constraints plus the active inequality constraints. As a result of the relationship, the zeros of the set of constraints in Equation 5.8 might be replaced by $(-\frac{1}{2}r\lambda_j)$, where the λ_j are approximations of the Lagrange multipliers; similarly the zeros of the set of constraints in Equation 5.7 might be replaced by $(-\frac{1}{2}r\lambda_j)$. Tests on problem sets indicate that this replacement improves the performance of the EQP algorithms. Consequently, the equality quadratic-programming sub problem for one stage ***k*** (the superscript ***k*** is suppressed) becomes:

$$\begin{array}{ll} \text{minimise} & s^T \nabla f(x) + \frac{1}{2} s^T B s \\ \text{subject to} & As = -\frac{r}{2}\lambda - \hat{h}(x) \end{array} \qquad (5.10)$$

where $\lambda = [\lambda \quad \cdots \quad \lambda_m]^T$, $x = [x_1 \quad \cdots \quad x_n]^T$, $s = [s_1 \quad \cdots \quad s_n]^T$, and A is the Jacobian matrix of the active constraints (the row $a_i = \nabla^T g_j$ or $\nabla^T h_j$). The vector of estimates of the Lagrange multipliers is given by the solution of m equations (equal to the number of active constraints):

$$\left(\frac{r}{2}I + AB^{-1}A^T\right)\lambda = AB^{-1}\nabla f(x) - \hat{h}(x) \tag{5.11}$$

and the solution can be written explicitly as:

$$s = B^{-1}\left[A^T\lambda - \nabla f(x)\right] \tag{5.12}$$

An especially useful feature of formulation of Equation 5.10 is that for $r > 0$, the Lagrange multipliers λ and the search direction s are defined even if linear dependence exists among the rows of A. The recursive EQP algorithm just requires for satisfactory search that r at each stage k satisfy the following inequality (Bartholomew-Biggs, 1982):

$$-\frac{r}{2}\hat{h}^T x(\lambda) - \hat{h}^T(x)\hat{h}(x) \le 0 \tag{5.13}$$

The inequality Equation 5.13 means that $\hat{h}(x)As \le 0$ because of the constraints in Equation 5.10, and therefore s is in a direction along which the active constraint violation is not increasing.

When the search direction has been established by solving the quadratic programming sub problem, a minimisation algorithm must be used to calculate a step size in the search direction. Various kinds of line searches have been employed in which a quadratic-loss penalty function is the objective function, others use the exact penalty function, while others use the Lagrange function or an approximation (Edgar and Himmelblau, 2001).

To ensure that the SQP method converges from remote starting points, it is common to use a merit function to control the size of the steps (in line search methods) or to determine whether a step is acceptable and whether the trust region radius needs to be modified (in trust region methods). It plays the role of the objective function in unconstrained optimisation, since each step provides a sufficient reduction in it.

Although the merit function is needed to induce global convergence, interfering with the "good" steps (those that make progress toward a solution) is not wanted.

Some merit functions can delay the rapid convergence performance of SQP methods by rejecting steps that make good progress toward a solution. This undesirable phenomenon is often called the Maratos effect (Nocedal and Wright. 1999). If no measures are taken, the Maratos effect can dramatically slow down SQP methods. Not only does it interfere with good steps away from the solution. but it can also prevent super-linear convergence from taking place.

Three important considerations when choosing an appropriate merit function for an algorithm are:

1. Choosing step lengths based on the merit function should lead to global convergence.
2. It should be possible to achieve sufficient decrease in the merit function given the search direction defined by the sub problem.
3. The merit function should not inhibit the rate of convergence.

Popular choices for the merit function with the SQP method include the absolute value merit function (which can be considered an exact penalty function) and the augmented Lagrangian merit function (created by augmenting the Lagrangian with a new term that disappears at x^*). Both of these use penalty parameters, and the choice of a particular penalty parameter can have a substantial effect on efficiency.

SQP methods, under suitable assumptions, show favourable local convergence properties. When B closely approximates $\nabla^2{}_x L(x,\lambda)$, the QP sub-problem generates the Newton direction (Goldsmith, 1999). If unit steps are taken and the method converges, a quadratic asymptotic rate of convergence arises. If approximations are used in place of exact second derivatives, the steps taken are no longer Newton steps. and the convergence rate may deteriorate. When an inexact Hessian approximations is used. the infeasibilities tend to zero faster than the reduced gradient converges to zero (meaning that feasibility is reached faster than optimality).

To assure a unique minimiser for the sub-problem, some SQP methods require the QP Hessian to be positive definite. This requirement interferes with the effective use of exact second derivatives. A further difficulty with exact second derivatives is that in many cases they are either unknown or too computational expensive. In practice the Hessian of the Lagrangian is often approximated from the first derivatives using a quasi-Newton approach that produces a positive-definite matrix.

Problems for real applications are often large, but sparse in terms of the number of variables appearing in any single constraint or the number of variables appearing in any single constraint or the number of constraints involving any single variable. Methods have been developed to reduce the storage and effort required to handle large-scale problems, including limited-memory methods, reduced Hessian approximations, sparse quasi-Newton approximations, and partial separability.

Most SQP algorithms have been developed for positive-definite Hessian approximations as previously defined. Nonetheless, some recent research allows indefinite Hessian approximations but terminates QP sub-problems at the first stationary point with a descent direction formed from negative multiplier estimates (Gill et al, 2002). Some other research by Goldsmith (1999) proposes disaggregation of the Hessian approximation, by means of individuals Hessians and form the weighted sum. This last approach is an application of the partial separability to quasi-Newton approximation of the Hessian of the Lagrangian; it has shown several advantages for sparse problems.

5.3 SQP IMPLEMENTATION IN MATLAB

In SQP, Newton's method for constrained optimisation is simulated just as it is done for unconstrained optimisation (Biggs, 1975 and Han, 1977). The MATLAB SQP implementation comprises three main stages, which are discussed in the Toolbox User's Guide (Coleman et al, 1999) and are briefly introduce in the next sub sections.

5.3.1. Updating The Hessian Matrix of the Lagrangian function

It is known that the quasi-Newton approach was originally developed for unconstrained and linearly constrained optimisation. Thus, for nonlinear constrained problems, the Hessian of a composite function including the objective function, constraint functions and Lagrange multipliers needs to be estimated. For this purpose, at each iteration, a positive definite quasi-Newton approximation of the Hessian, H, is updated using the BFGS method (Powell, 1978) mentioned in the previous section. It can be seen that this update depends on the gradient difference of the Lagrangian q_k, which is a function of x_k; in addition, λ_i is an estimate of the Lagrange multipliers and s_k is the search direction.

$$H_{k+1} = H_k + \frac{q_k q_k^{\,T}}{q_k^{\,T} s_k} - \frac{H_k^{\,T} s_k^{\,T} s_k H_k}{s_k^{\,T} H_k s_k} \tag{5.14}$$

where

$$s_k = x_{k+1} - x_k$$

$$q_k = \left(\nabla f(x_{k+1}) + \sum_{i=1}^{m} \lambda_i \cdot \nabla g_i(x_{k+1}) \right) - \left(\nabla f(x_k) + \sum_{i=1}^{m} \lambda_i \cdot \nabla g_i(x_k) \right)$$

To calculate q_k accurately, it is necessary to decide which multiplier estimates to use at each iteration. Powell (1978) emphasized the importance of maintaining positive definiteness in quasi-Newton methods for constrained optimisation even if it may be positive indefinite at the solution point. With this in mind, a positive definite Hessian is maintained providing $q_k^T s_k$ is positive at each update and that H is initialised with a positive definite matrix. Conversely, when $q_k^T s_k$ is not positive, q_k is modified on an element by element basis so that $q_k^T s_k > 0$.

Accordingly, for the initial stage of the variation, the most negative element of $q_k s_k$ is continually halved. This course of action carries on until $q_k{}^T s_k$ is greater than or equal to a set value of $1\text{x}10^{-5}$ (considered to be a sufficiently small parameter). The second stage initiates if after this method, $q_k{}^T s_k$ is still not positive. Then, q_k is modified by adding a vector v multiplied to a constant scalar w that is analytically increased until $q_k{}^T s_k$ becomes positive. As a result, the following equations can be formulated:

$$q_k = q_k + wv \tag{5.15}$$

$$\text{if } (q_k)_i \cdot w < 0 \text{ and } (q_k)_i \cdot (s_k)_i < 0 \quad (i = 1, \ldots, m)$$

$$\text{then } v_i = \nabla g_i(x_{k+1}) \cdot g_i(x_{k+1}) - \nabla g_i(x_k) \cdot g_i(x_k)$$

$$\text{or else } v_i = 0$$

The functions in MATLAB that use SQP (in Version 5.2: *fmincon*, *fminimax*, *fgoalattain* and *fseminf*) and thus this procedure for updating the Hessian matrix; can display information on whether the Hessian has to be modified using the first or the second stage of the procedure to maintain it positive definite (by displaying *Hessian modified* and/or *Hessian modified twice* respectively). It is also possible to obtain a message of *no update* meaning only that $q_k{}^T s_k$ is almost zero.

5.3.2. Quadratic Programming Problem Solution

The general Nonlinear Optimisation Problem (NLP) from Equation 5.2 can now been simplified as an Inequality constrained Quadratic Program (IQP) (considering bound constraints expressed as part of the inequality constraints). Once more L is the Lagrangian in the variables x, λ_i are the Lagrange multipliers, f and $g_1, \cdots, g_m$, are all defined functions and m is the total number of constraints.

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i \cdot g_i(x) \tag{5.16}$$

Herewith, this approximation is used to generate a QP sub problem whose solution is used to form a search direction for a line search procedure. The main idea is the formulation of a QP sub problem based on a quadratic approximation of the Lagrangian function.

By means of linearising the non-linear constraints the QP sub problem can then be attained. This QP sub problem can be defined as follows:

$$\text{minimise} \quad \frac{1}{2} d^T H_k d + \nabla f(x_k)^T d \quad \text{which can be rewritten as}$$

$$\text{minimise} \quad q(d) = \frac{1}{2} d^T H d + c^T d \tag{5.17}$$

$$\text{subject to} \quad A_i d = b_i \qquad i = 1,\ldots,m_e$$

$$A_i d \leq b_i \qquad i = m_e + 1,\ldots,m$$

where $d \in \Re^n$ and is the search direction from the QP sub problem, H_k is the positive definite approximation of the Hessian matrix of the Lagrangian function and $c=\nabla f(x_k)$ is the gradient of the function f evaluated at x_k. A_i is an element of matrix A which is an estimate of the active constraints and b_i are the constraint boundaries. Ultimately, m_e is the number of equality constraints and m is the total number of constraints (equality and inequality constraints collectively).

The method used in MATLAB's Optimization Toolbox is the projection method similar to the one described by Gill et al (1991). This active set strategy has been modified for both Linear Programming (LP) and QP problems. The also called null-space active set methods are an efficient way of solving positive definite QP problems when the dimension of the null-space is not too large (≤ 1000). It can be described as follows:

(i) A linear programming problem is solved to determine an initial feasible point. If the current point from the SQP method is not feasible, then a point can be found by solving the linear programming problem:

$$\begin{aligned} &\text{minimise} \quad \gamma \\ &A_i x = b_i \qquad\qquad i = 1,\ldots,m_e \\ &A_i x - \gamma \le b_i \qquad\qquad i = m_e + 1,\ldots,m \end{aligned} \tag{5.18}$$

where $\gamma \in \Re$, $x \in \Re^n$. A feasible point (if one exists) to the previous equation can be found by setting x to a value that satisfies the equality constraints. This can be achieved by solving an under-or-over determined set of linear equations formed from the set of equality constraints. If there is a solution to this problem, then the slack variable γ is set to the maximum inequality constraint at this point.

(ii) An iterative process is then started whereby at each iteration, a step is taken from the current position toward the optimal solution. With this technique an active set $\overline{A}_k$ is kept, which is an estimate of the active constraints and bounds at the solution point. This active set is updated at each iteration, k, and this is used to form a basis for a search direction $\hat{d}_k$. Variables whose bounds are in the active set are called fixed and the others are called free. The method proceeds to move to a constrained stationary point of the QP by holding the fixed set of variables constant and temporarily ignoring the other bounds.

(iii) In each of these iterations, the search direction $\hat{d}_k$ is calculated to minimise the cost function while remaining within the active constraint boundaries. The feasible subspace for $\hat{d}_k$ is calculated from a Z_k which is basis for the null space of the rows of the active set $\overline{A}_k$. In consequence, a search direction which is formed from linear summation of any combination of the columns of Z_k, is guaranteed to remain on the boundaries of the active constraints. Having found Z_k, a new search direction $\hat{d}_k$ is sought that minimises $q(d)$ where $\hat{d}_k$ is in the null space of the active constraints, that is, $\hat{d}_k$ is a linear combination of the columns of $Z_k : \hat{d}_k = Z_k p$ for some vector p. The quadratic can be seen as a function of p substituting for $\hat{d}_k$ in Equation 5.17, so that:

$$q(p) = \frac{1}{2} p^T Z_k{}^T H Z_k p + c^T Z_k p \qquad (5.19)$$

This equation can differentiated with respect to p to obtain:

$$\nabla q(p) = Z_k{}^T H Z_k p + Z_k{}^T c \qquad (5.20)$$

In this Equation 5.20 $\nabla q(p)$ is the projected gradient of the quadratic function because it is the gradient projected in the subspace define by Z_k. The term $Z_k{}^T H Z_k$ is then called the projected Hessian. Subsequently, the minimum of the function $q(p)$ in the subspace defined by Z_k occurs when $\nabla q(p) = 0$, which is the solution of the system of linear equations.

(iv) The active set of constraints is updated at every iteration. A step is then taken according to the following formula:

$$x_{k+1} = x_k + \alpha \hat{d}_k \qquad \text{where } \hat{d}_k = Z_k{}^T p \qquad (5.21)$$

Because of the quadratic nature of the objective function, at every iteration there are only two choices of step length α. If the solution of $\hat{d}_k$ to the EQP problem in equation 5.17 is feasible with respect to all the bounds, the full step of length one is taken and a stationary point is reached for the QP. If not, the maximum feasible step with respect to the bounds is taken along the search direction $\hat{d}_k$. This sequence repeats until the full step is taken. At the stationary point, if for any bound there exists a negative Lagrange multiplier, the associated bound is dropped from the working set and the procedure starts over. If all the multipliers are positive, then the algorithm stops. On the other hand, if the QP is feasible and has no degenerate vertices, this process terminates in a set number of iterations.

5.3.3. Line Search and Merit Function Calculation

Line search methods attempt to maintain a fast local convergence by limiting the size of the step taken from the current point to the next iterate (see Equation 5.21). Nevertheless, maintaining feasibility at every iteration becomes difficult for nonlinear constraints and it is not inmediately obvious how to choose the step length parameter α_k. The aim is that the next iterate minimises the objective function and also reduces the infeasibilities of the constraints. To ensure that improvement is made towards the solution, a merit function can be used to measure whether one point is better than another. In MATLAB, a merit function $\psi(x)$ used by Han (1977) and Powell (1978) has been implemented and it is defined by:

$$\psi(x) = f(x) + \sum_{i=1}^{m_e} r_i \cdot g_i(x) + \sum_{i=m_e+1}^{m} r_i \cdot \max\{0, g_i(x)\} \qquad (5.22)$$

Then the penalty parameter r_i can be calculated as:

$$r_i = (r_{k+1})_i = \max_i \left\{ \lambda_i, \frac{1}{2}\left((r_k)_i + \lambda_i\right) \right\}, \qquad (5.23)$$

This penalty parameter allows positive contribution from constraints that are inactive in the QP solution but were recently active. For this purpose, the penalty parameter r_i is initially set using the Euclidean norm $\|\cdot\|$ as follows:

$$r_i = \left\| \frac{\nabla f(x)}{\nabla g_i(x)} \right\| \qquad (5.24)$$

Thus, larger contributions to the penalty parameter from constraints with smaller gradients can be assured.

5.4 SQP FOR OPTIMAL CONTROL OF BEER FERMENTATION

The selected beer fermentation mathematical process modelled in SIMULINK, has been modified in order to be part of the optimisation using SQP with the help of the Optimisation Toolbox from the MATLAB Software (Coleman et al, 1999). It is important to note that MATLAB's Optimisation Toolbox offers a range of possibilities for optimal control. Other optimisation methods have been implemented by using different functions from the Optimisation Toolbox and consequently changing the mathematical model to incorporate the particular algorithm requirements. This has been done in order to optimise the fermentation process using an appropriate and beneficial technique.

In order to deal with the bound constraints that the beer fermentation problem requires, a simple transformation mentioned by Noton (1972) has been used. This in order to convert from a constrained to an unconstrained optimisation problem and thus, being able to use the functions from the MATLAB Software. Considering simple boundaries of the form:

$$L_i \leq x_i \leq U_i$$

where L_i and U_i are vectors that include the lower and upper boundaries respectively and x_i is the vector of variables to be optimised.

These boundaries can now be included into a new optimisation variable $\tilde{x}_i$ using the following equation:

$$x_i = L_i + (U_i - L_i)\left(1 - e^{-\tilde{x}_i^2}\right)$$

This optimisation variable has been included in the optimisation routine (script file) and the optimisation of an unconstrained problem can be pursued.

Herewith, two different script files have been created to include the unconstrained nonlinear maximisation functions: ***fminsearch*** and ***fminunc***. The *fminsearch*

function, attempts to find the maximum of an unconstrained multivariable function starting at an initial estimate, by means of the simplex search method (does not use numerical or analytical gradients). The next function *fminunc* uses an initial estimate too, but also requires the gradient of the objective function to be defined: this is a subspace trust region method based on the interior-reflective Newton method. Nonetheless, the experimentation with these functions did not provide any satisfactory results so it was disregarded as an appropriate tool for optimising the fermentation model simulated. Presumably the reason for this due to the high non linearities of the process equations and the algorithms not been suitable for this process optimisation. With this, the computer always presented an error after just a few iterations of the optimisation run.

Another MATLAB function for scalar nonlinear minimisation with bounds, ***fminbnd*** (also used in Chapter 3), has been tested for the optimisation of the complete problem. This function aims to find the minimum of a function of one variable on a fixed interval; for this purpose upper and lower bundaries of 0 and 16°C have been included. Once again, no satisfactory results were obtained using this technique. The complete justification of this behaviour is still not completely understood but the use of these other functions was stopped since the results using the SQP function ***fmincon*** were very optimistic.

Subsequently, a MATLAB script file (m-file) that includes the necessary instructions for the optimisation algorithm to identify the objective function and also that obtains the final objective value is created. The simulated model includes the five terms of the objective function to be maximised. Lower and upper boundaries of 0 and 16°C correspondingly have been set for the temperature values. The preset duration of the fermentation process formulates the equality constraints: initial time t_0 (0 hours) and final time t_f (160 hours). The remaining inequality constraints have been added so the coordinate points for the time values are always greater than the previous one but smaller than the next one. To achieve this, three intermediate times (t_1, t_2 and t_3) have been set in order to be found by the SQP optimisation routine. The reason being that a smooth profile is pursued and selecting three parameters has been a sufficient choice in earlier chapters.

maximise

$$J = 10 \cdot e_{end} - 5.73 \times 10^{-8} \cdot e^{(95 \cdot diac_{end})} - 1.16 \times 10^{-29} \cdot e^{(4.6 \cdot acet_{end})} - \int_0^{t_f} 9.91 \times 10^{-7} \cdot e^{(2.31 \cdot T)} dt - \sum_{i=1}^{160} \frac{|T_{i-1} - T|}{\Delta t}$$

subject to

$$t_0 = 0 \qquad t_f = 160$$

$$t_0 < t_1 \qquad t_1 < t_2 \qquad t_2 < t_3 \qquad t_3 < t_f$$

The SIMULINK model includes an input port that allows the optimisation algorithm to interact and change the input temperature profile; thus, reaching a maximum value of the objective function that is already included in the SIMULINK model. It is important to note that interpolation between coordinate points can be selected inside this input port. Figure 5.1 shows this model.

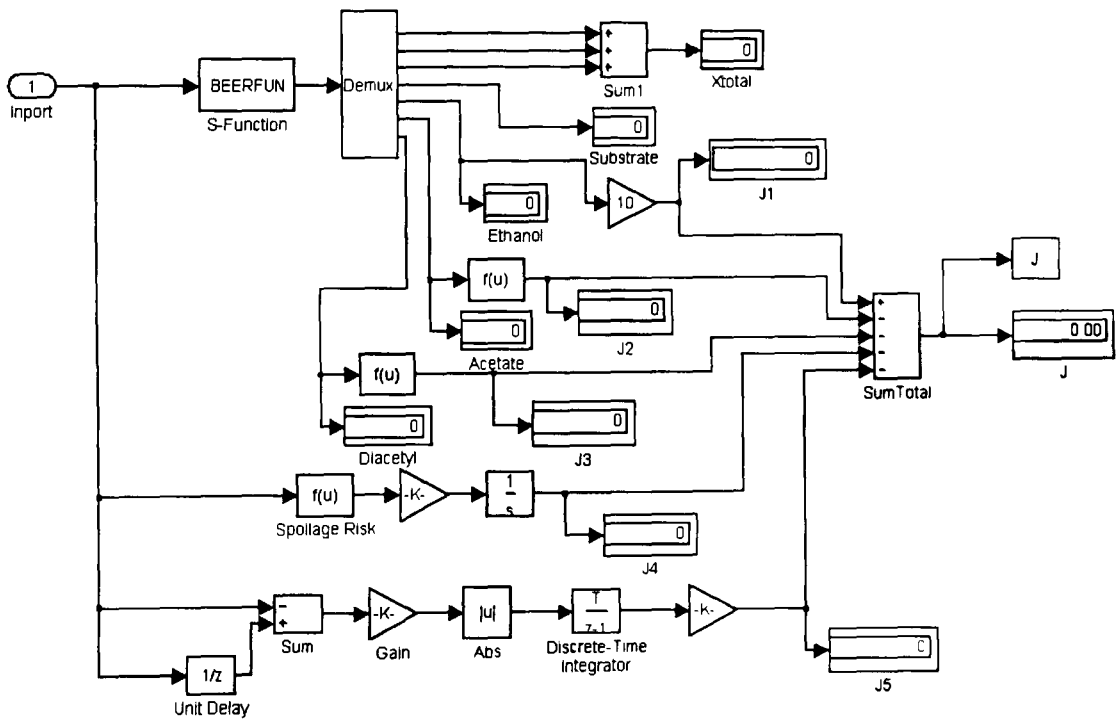

Figure 5.1 SIMULINK model used for the SQP optimisation

Subsequently, a new MATLAB script file called "simfunsqp.m" that includes the necessary instructions for the optimisation algorithm to identify the objective function and also obtains the final objective value is created. Another m-file "beersqp.m" including the optimisation script, some initial and function parameters is also formed; this file is the responsible to call the "fmincon.m" m-file from MATLAB's Optimization Toolbox that performs the SQP optimisation and calls other required mathematical functions/resources.

Different tests have been performed in order to see the effect of changing the default parameters of the minimisation function to the output variables of the fermentation process. Table 5.2 shows the default values used by the SQP algorithm under the MATLAB toolbox.

Parameters	Value	Description
MaxIter	400	Maximum number of iterations.
TolCon	$1x10^{-6}$	Termination tolerance on the constraint violation.
TolFun	$1x10^{-6}$	Termination tolerance on the function value.
TolPCG	0.1	Termination tolerance on the PCG iteration.
TolX	$1x10^{-6}$	Termination tolerance on X.
DiffMaxChange	$1x10^{-1}$	Maximum change in variables for finite difference gradients.
DiffMinChange	$1x10^{-8}$	Minimum change in variables for finite difference gradients.

Table 5.2 Default parameters used by the Optimisation Toolbox

Some of the most relevant parameters needed for the optimisation function have been changed from its original values in the SQP algorithm after some experimentation. These new values included in Table 5.3, provide a solution that can be reached in less time and still maintain a good performance value of the objective cost function.

Parameters	Value	Description
LB	0	Lower boundary vector for the temperature profile and time.
UB	16, 160	Upper boundary vectors for the temperature profile and time.
MaxIter	100	Maximum number of iterations.
TolCon	$1x10^{-4}$	Termination tolerance on the constraint violation.
TolFun	$1x10^{-4}$	Termination tolerance on the function value.
TolPCG	0.01	Termination tolerance on the PCG iteration.
TolX	$1x10^{-4}$	Termination tolerance on X.
DiffMaxChange	$1x10^{-1}$	Maximum change in variables for finite difference gradients.
DiffMinChange	$1x10^{-5}$	Minimum change in variables for finite difference gradients.

Table 5.3 Changed parameters used for the SQP optimisation

In order to obtain a better optimisation with SQP and after several tests with different combinations of parameters, five temperature values corresponding to five different times have been set to be optimised. The first and the fifth of these time parameters have been fixed to the initial and final times of the fermentation process (0 and 160 respectively), and also used as boundaries for the remaining time values.

Ten initial values have been selected as the initial guess for the optimisation algorithm. The industry's temperature profile has been chosen for the first test; hereafter, different constant temperature profiles have been used.

In order to show a summary of the optimisation and simulation with SQP, the following Table 5.4 is presented; it includes respectively: the Case examined (Industrial Profile first and then constant temperature values), the total amount of iterations required for the optimisation, the final cost function J value obtained, the final step size used in the last iteration and the total computational time required for the optimisation.

With these results it can be said that even without any previous knowledge of the simulated process itself, by means of an assumed initial temperature profile; the SQP algorithm performs a satisfactory optimisation of the process reaching a good level of maximisation in not many iterations. The computational time used varies depending on the first guess included for the algorithm, however, the optimisation still achieves approximately the same performance index value at the end regardless of which initial temperature profile have been used.

Case	Total Iterations	J value	Final Step size	Computational Time required*
1 (I.P.)	36	599.207	0.5	3.009
2 (3°C)	59	599.632	1	4.654
3 (4°C)	56	600.2	1	3.968
4 (7°C)	45	600.199	0.0156	3.338
5 (8°C)	71	600.201	1	5.036
6 (9°C)	28	600.204	1	2.047

7 (11°C)	29	600.204	1	2.198
8 (12°C)	34	600.201	1	2.513
9 (14°C)	40	600.197	1	3.303
10 (16°C)	53	600.194	-6.10E-05	4.049

*Computational time in minutes using an 800 MHz CPU speed and 256MB RAM PC

Table 5.4 Results of the optimisation with SQP for Cases 1-10

The next results are a review obtained for the best situation considered, Case #6:

The initial temperature profile has again been set constant to 9° Centigrade along the entire fermentation process and interpolation is required to obtained the optimised profile. Figure 5.2 illustrates these temperature profiles.

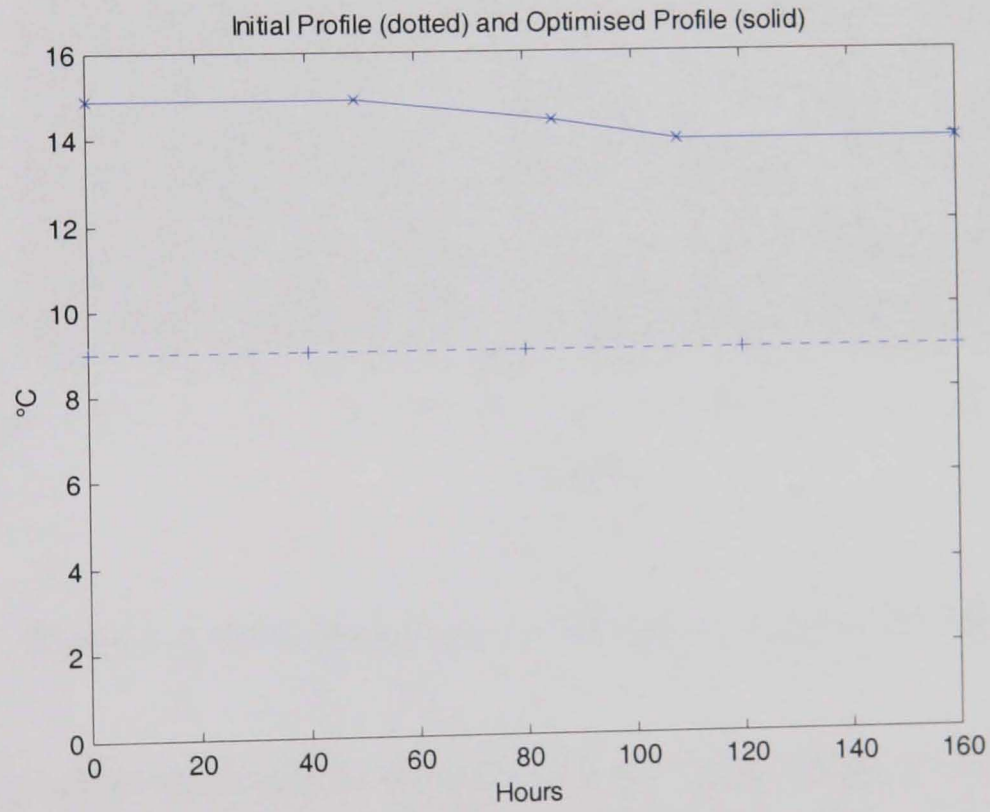

Figure 5.2 Initial and Optimised Temperature Profiles for Case #6

The final results of the optimisation are included in Table 5.5:

Iter	F-count	f(x)	Max. constr.	Step-size	Direc. deriv.	Procedure
28	387	-600.204	0.000E+00	1	-0.000094	

Table 5.5 Results from the optimisation of the Case #6

In this occasion, the Hessian was modified once only in one instance during iteration 26th during the optimisation of Case #6; the maximum value for the directional derivative reached was -326 at the 1st iteration and six different step-sizes were used along the entire optimisation, with the final one being 1.

The development of the SQP algorithm shown in Figure 5.3 includes the iteration number (Iter) versus the objective function value (f(x)).

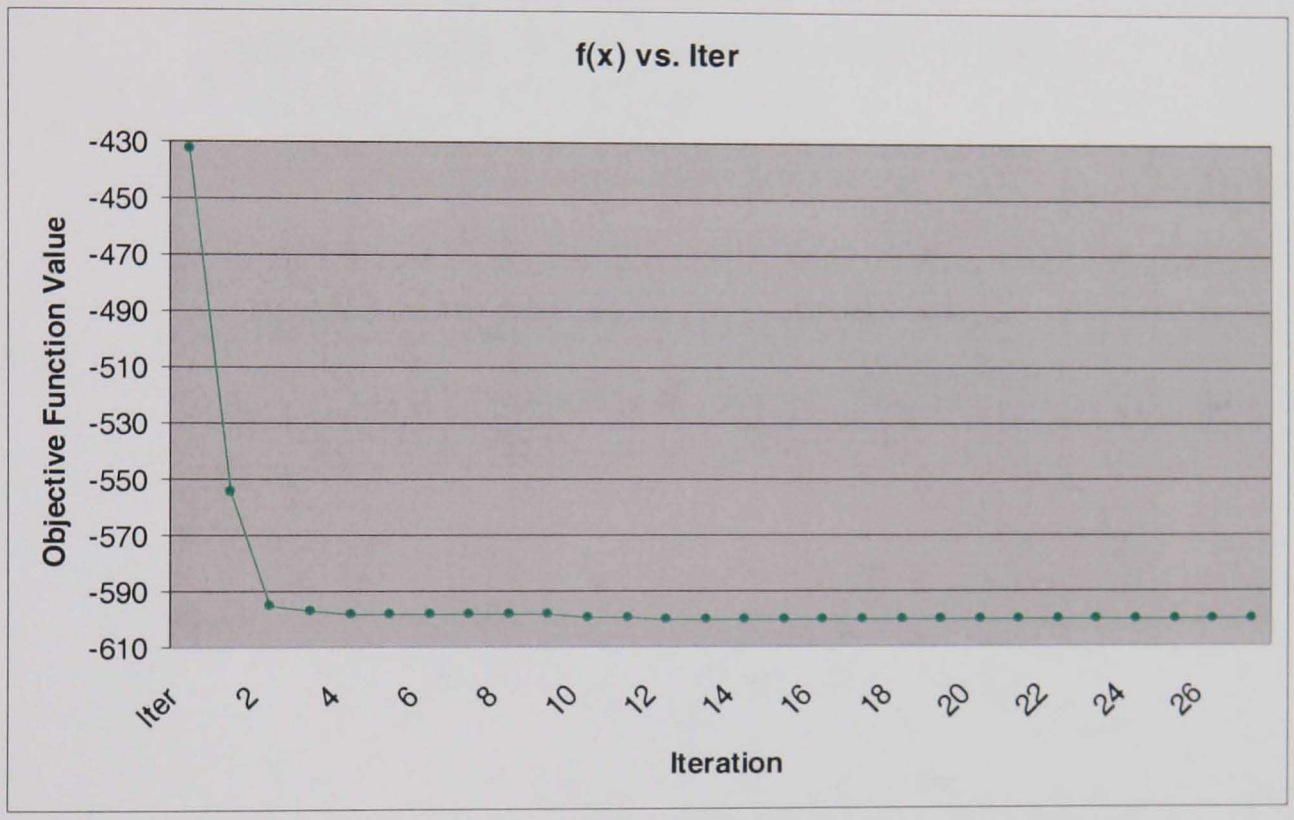

Figure 5.3 Development of the SQP algorithm for Case #6

With the new temperature profile applied to the simulated model of the fermentation process, the behaviour of the differential equations' parameters can be observed in Figures 5.4-5.7.

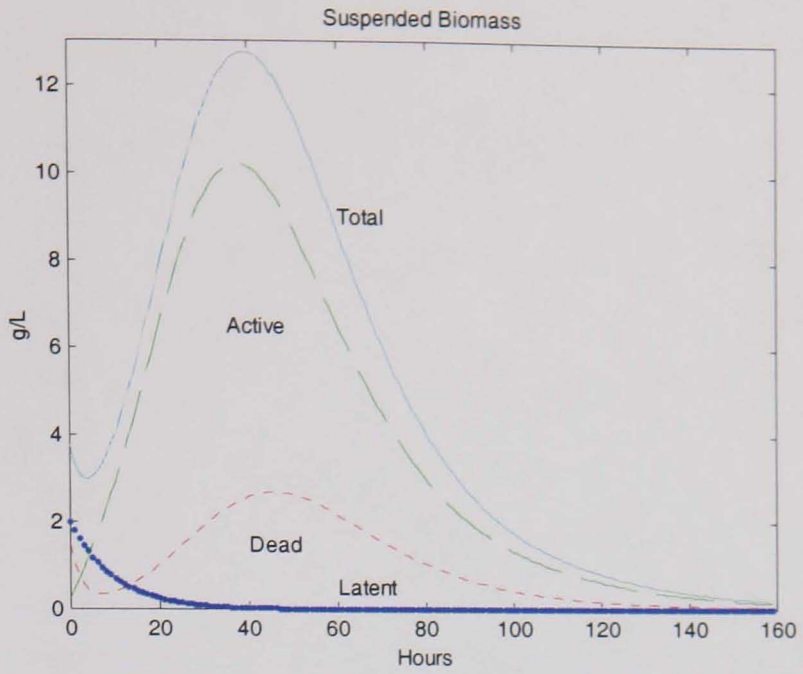

Figure 5.4 Suspended Biomass behaviour for Case #6

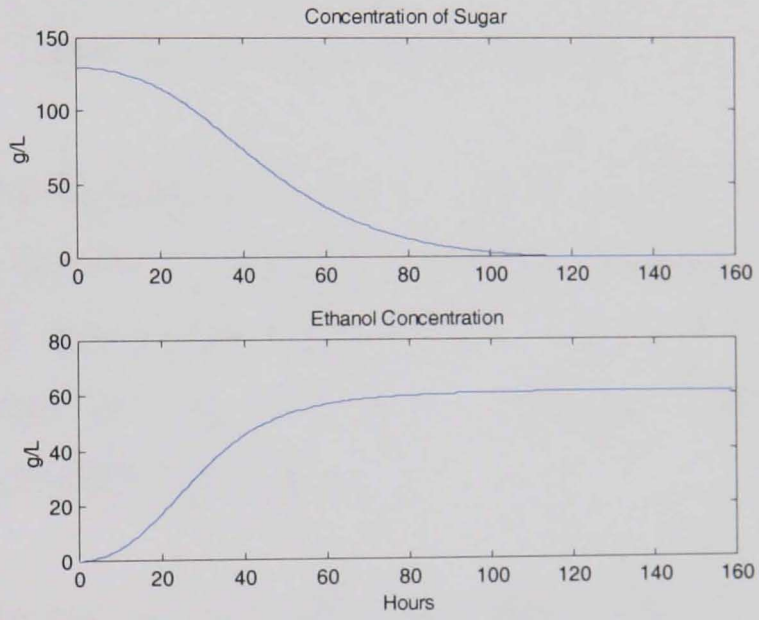

Figure 5.5 Sugar and Ethanol Concentration for Case #6

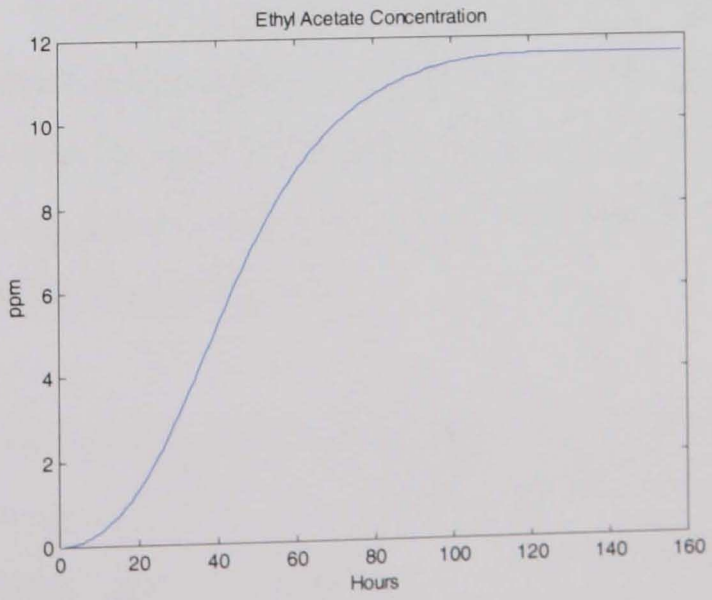

Figure 5.6 Ethyl Acetate Concentration for Case #6

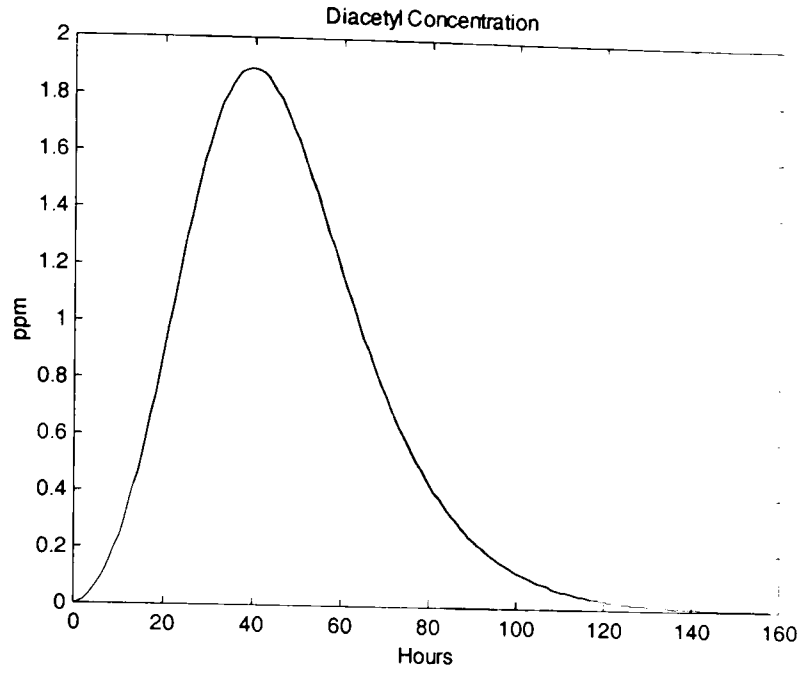

Figure 5.7 Diacetyl Concentration for Case #6

With the industrial temperature profile a maximum objective function value of 518.90 has been attained in chapter 2; this value compared with the maximum achieved after the optimisation with the SQP algorithm of 600.204, gives an increment in excess of about 16% in the objective function value after the optimisation using the SQP algorithm.

After these results and knowing that the initial profile for Case #6 (a constant temperature of 9°C) has demonstrated to be a suitable initial guess for the SQP algorithm; the same procedure can be carried out using different number of intermediate points. Seven new cases have been considered: zero, one, two, four, five, six and seven intermediate points (in that order). It can be said that more than seven intermediate points has not been considered given that the temperature profile obtained would be very discontinuous to be implemented in practice and hence no improvement is expected from this.

A summary of the optimisation results for Cases A-G using the SQP algorithm can be seen in the following table 5.6; it includes respectively: the Case examined, the number of intermediate points required (to produce the inequality constraints considered), the total amount of iterations used in the optimisation, the final cost

function J value obtained, the final step size used in the last iteration and the total computational time for the optimisation.

Case	Intermediate Points	Total Iterations	J value	Final Step size	Computational Time Required*
A	0	8	599.174	1	0.363
B	1	21	599.875	1	1.426
C	2	33	600.201	0.5	2.513
D	4	45	600.201	1	5.145
E	5	40	600.203	-6.10E-05	5.455
F	6	53	600.204	1	7.964
G	7	53	600.204	0.125	6.914

*Computational time in minutes using an 800 MHz CPU speed and 256MB RAM PC

Table 5.6 Results of the optimisation with SQP for Cases A-G

Together with this, the best results obtained with case F have been included as follows:

For Case F the initial temperature profile has once again been set constant to 9°C along the entire fermentation. Nonetheless, in this case six intermediate points are needed. Figure 5.8 shows the initial and final temperature profiles.

The final results of the optimisation are included in Table 5.7:

Iter	F-count	f(x)	Max. constr.	Step-size	Direc. deriv.	Procedure
53	1088	-600.204	0.000E+00	1	-0.000031	Hessian modified

Table 5.7 Results from the optimisation of the Case F

For Case F, the Hessian was modified once in four occasions during the entire optimisation (iterations 41st, 50th, 51st and 53rd); the maximum value reached by the directional derivative was -367 at the 1st iteration and a total of ten different step-sizes were used along the entire optimisation, including the final value 1.

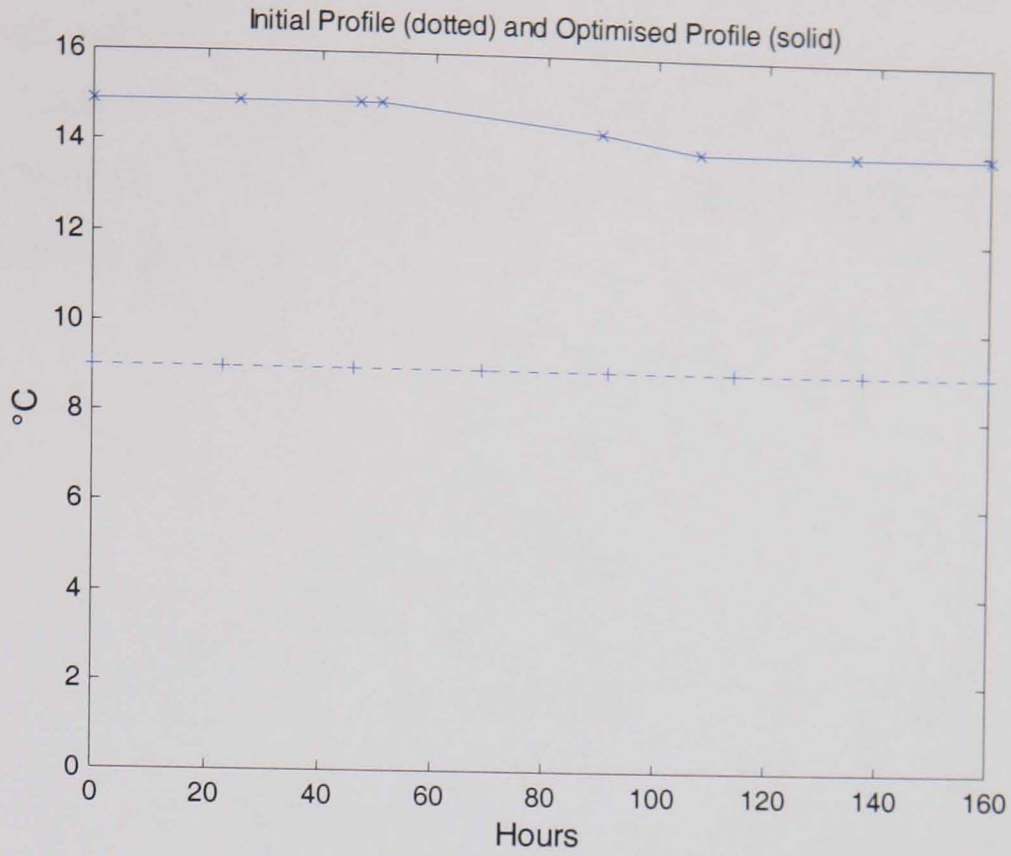

Figure 5.8 Initial and Optimised Temperature Profiles for Case F

The development of the SQP algorithm shown in Figure 5.9 includes the iteration number (Iter) versus the objective function value (f(x)).

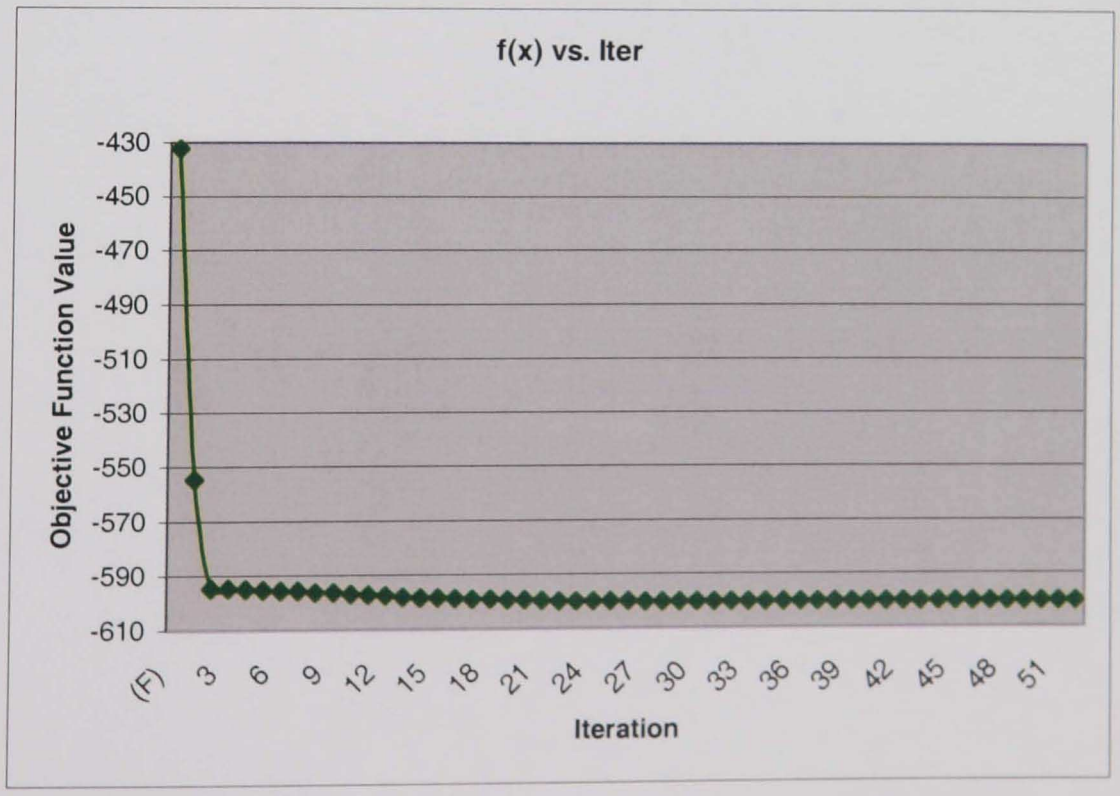

Figure 5.9 Development of the SQP algorithm for Case F

It can be seen from these results that the best optimised temperature profile is still Case #6 from Table 5.4. This due to the fact that although the maximum J value reached in Table 5.6 is equivalent to the previous value obtained, the temperature profile itself cannot easily be implemented in practice because of the uneven characteristics that it presents.

5.5 SUMMARY

The SQP Toolbox from MATLAB which makes use of the Sequential Quadratic Programming Optimisation Algorithm, has proved to be a practical and useful tool for the optimisation of the mathematical model of the beer fermentation process selected. The results obtained in this chapter with the SQP optimisation, have surpassed the perfomance index valuess and computational time previously achieved with the other optimisation procedures.

The optimised temperature profiles obtained with the Sequential Quadratic Programming algorithm are suitable for implementation in practice according to previous knowledge in the literature. The computational time used by the optimisation algorithm varies depending on the first guess used for the algorithm, however, the SQP algorithm still reaches approximately the same J value at the end despite the initial temperature profile applied.

Further work can to be carried out using different non-linear optimisation techniques to see the feasibility of similar and/or more advanced algorithms to solve this or similar mathematical models of fermentation processes.

CHAPTER VI

FINAL REMARKS

Optimal control techniques and different optimisation algorithms for batch beer fermentation processes are the motivation of this thesis.

Fermentation processes in general have been examined in chapter 1 but focus has been made to beer fermentation and the factors that affect this kind of procedure. A comprehensive foreword in relation to the by-products of beer fermentation has also been included. A brief introduction about the history of beer control has been reviewed as well, trying to notice its progress with time. After that, old and new optimisation algorithms have been presented with detailed explanation of the methods and techniques.

Some approaches been used to model different kind of fermentation processes have been included in chapter 2. Together with this, a few basic mathematical equations aiming to model the fermentation behaviour have been included in order to show how simple or complicated fermentation processes could be. Hereafter, emphasis has been made on beer fermentation for batch processes; and the typical relationships that are part of basic modelling approximations are presented. The use of mathematical models for real fermentation processes and the different ways to implement them has been incorporated. The kinetic model developed by Andres-Toro et al (1998) has been introduced and simulated. Some parameters and equations have been taken from the original research and some others have been changed in order to obtain effective performance. Thus, this model has been replicated in SIMULINK under the MATLAB environment to be part of the optimisation techniques to follow in the next chapters. Simulation results have been compared against the original/real process state responses in order to obtain a model that behaves as accurately as possible following a real estimate of the industrial fermentation process.

With the help of the basic principles of the DISOPE algorithm, originally from work by Roberts (1993), and some other modifications made to adjust it to the simulated process

the optimisation of the process has been pursued in chapter 3. The Minimum Principle has been applied to a reduced situation of the fermentation process in order to obtain the Hamiltonian as an alternative for optimal control. An associated technique, the Gradient Method in Function Space, which is essentially a direct method of minimisation/maximisation of the Hamiltonian, has been applied. The maximisation of the defined objective function, for three different cases has lastly been presented using these two optimisation algorithms.

The Genetic Algorithms as a stochastic technique have proved to be suitable in the optimisation of fermentation processes and no previous knowledge, such an initial temperature profile, has been necessary to obtain a satisfactory result. The SIMULINK implementation described in chapter 4 is a flexible way to represent the mathematical model and was easy to interface with the Genetic Algorithm Toolbox by Chipperfield et al (1999). With this, a superior cost value function and a softer profile have been obtained by means of calculating average temperature values between specific time steps; thus making the results suitable for practical implementation. Hereafter, a decrease in the number of the decision variables for the optimisation algorithm has meant a reduction in the computational cost and has also made the resulting temperature profile useful for industrial applications.

The optimised temperature profiles obtained with the Sequential Quadratic Programming algorithm in chapter 5 are considered to be suitable for implementation in practice, without much time required for the optimisation procedure. Herewith, the computational time used by the optimisation algorithm varies depending on the first guess used for the algorithm. Nonetheless, the SQP algorithm reaches approximately the same performance index value J at the end, despite the initial temperature profile applied, and also incorporates the overall maximum value achieved if compared to the other optimisation techniques.

6.1 COMPARISON OF THE METHODS REVIEWED

With all these in mind, a more exhaustive comparison of these techniques for the optimisation of the beer fermentation process selected is pursued. As a starting point, a summary of the best results obtained with each optimisation algorithm have been integrated in Table 6.1. With this, the temperature profiles for each technique can be seen in Figures 6.1-6.6 and Table 6.2 has been created.

Table 6.1 shows a summary of the results obtained with the different optimisation techniques considered in the thesis. Only the parameters that may have a strong influence in the selection criteria have been included in the table.

The first column comprises the case considered within the specific Chapter (second column); then, the optimisation algorithm used for the specific case is in the third column; the number of decision variables can be seen in column four; the total amount of iterations required for the particular algorithm is incorporated in column five; column six confirms the maximum performance index (J) value reached and finally, column seven presents the total time (in minutes) required by each algorithm to reach the optimised profile.

Case	Included in Chapter	Optimisation algorithm	Decision variables	Iterations	*J* reached	Total time*
I.P.	*2*	*N/A*	*5*		*518.90*	
	3	Golden Section Search	1	15	597.808	0.0686
1	3	Gradient Method in Function Space	160	48	599.01	0.6528
A	3	DISOPE	160	65	598.87	0.6905
J	4	GAs	5	150	600.13	444.007
6	5	SQP	5	28	600.204	2.047

*In minutes using an 800MHz CPU PC with 256MB RAM

Table 6.1 Summarised results with the optimisation algorithms considered

Figures 6.1 to 6.6 show the final six temperature profiles that have been applied to the fermentation process modelled. The same temperature and time scales have been used to facilitate the assessment of the profile obtained and its value in practice for industrial purposes.

It is important to note that any judgment for the selection of the most suitable of these temperature profiles is very subjective and cannot be considered absolutely right for this same reason. The decision is based on the personal criteria of the control engineer and also in this case the previous knowledge of the industrial requirements in practice.

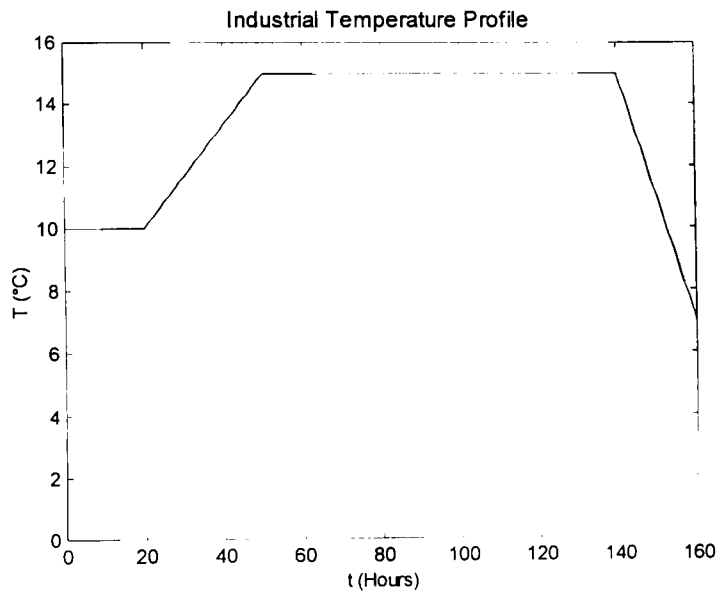

Figure 6.1 Industrial Temperature profile used in practice

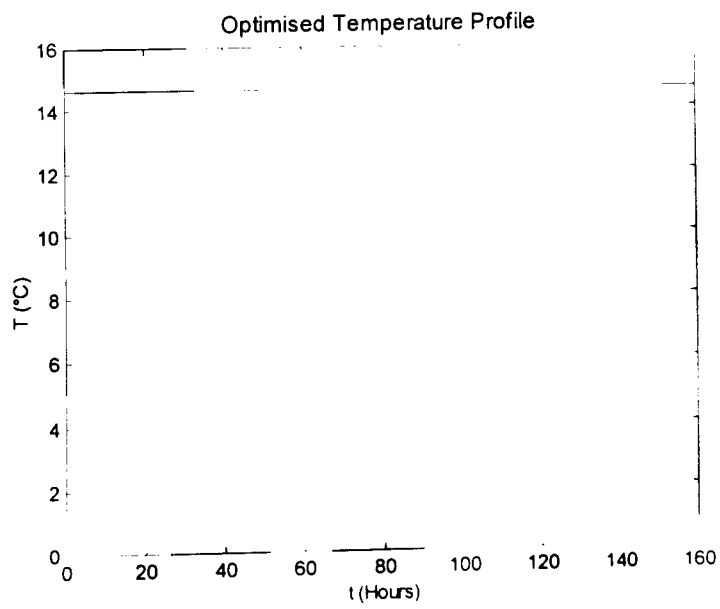

Figure 6.2 Temperature profile obtained with the Golden Section Search method

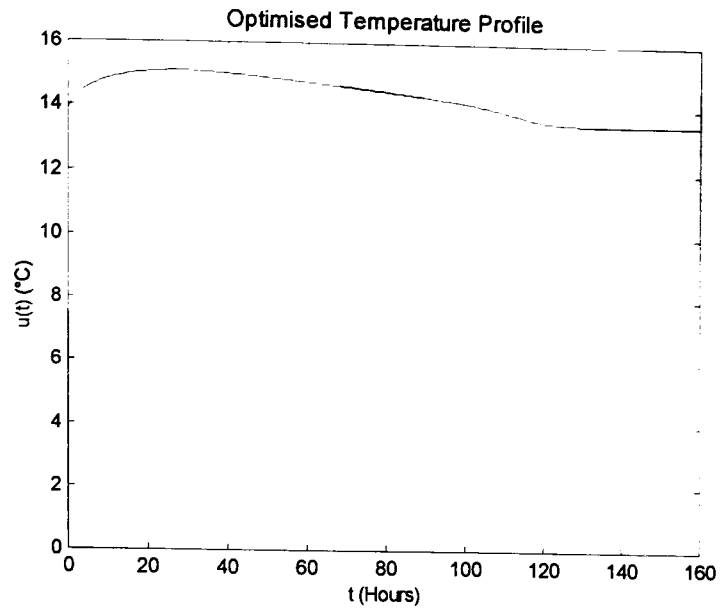

Figure 6.3 Temperature profile obtained with the Gradient Method in Function Space

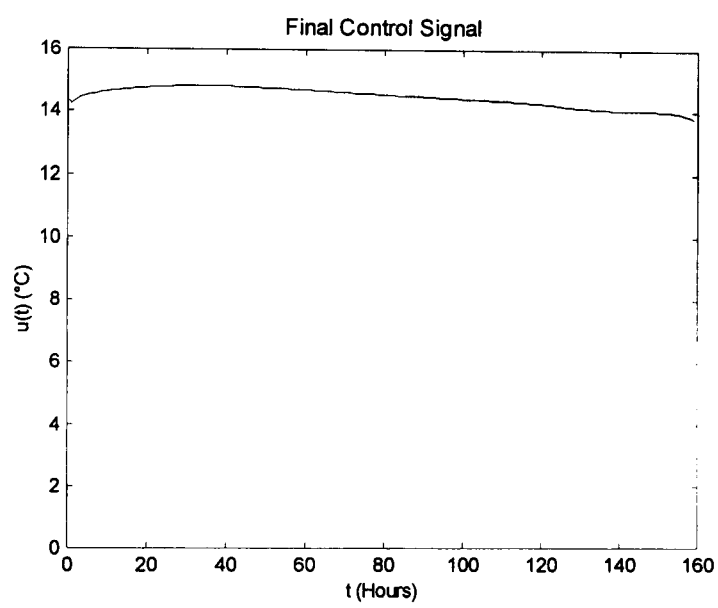

Figure 6.4 Temperature profile obtained with the DISOPE Algorithm

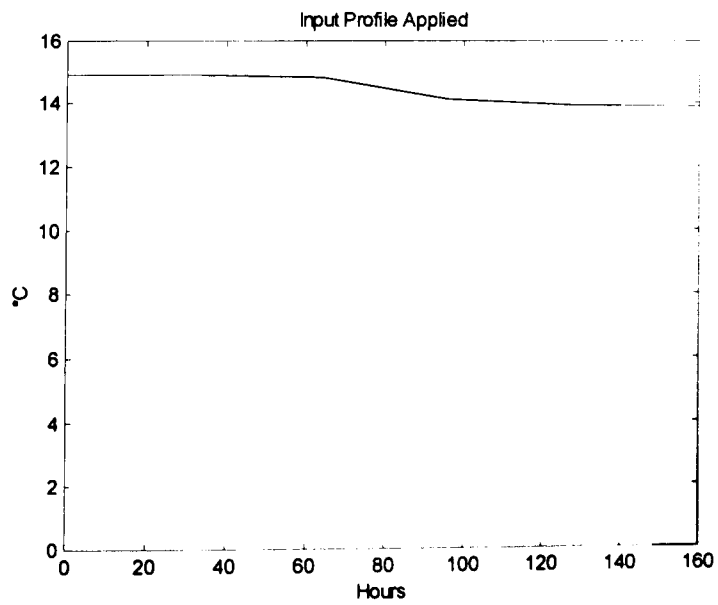

Figure 6.5 Temperature profile obtained with Genetic Algorithms

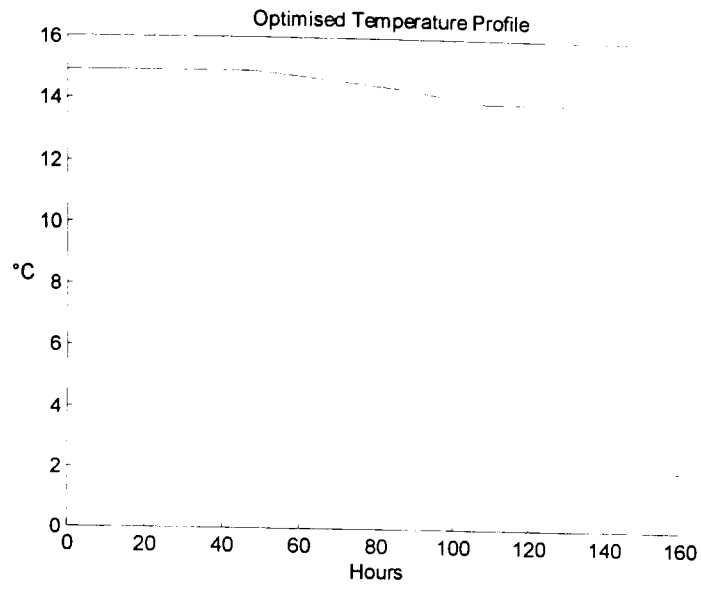

Figure 6.6 Temperature profile obtained with Sequential Quadratic Programming

With the intention of comparing the different optimisation techniques and by means of measuring and assessing the results obtained; different criteria used for selecting a valuable optimisation algorithm for the beer fermentation process have been considered. The main five factors include the following:

1. *Cost function value obtained*: the total value of the objective function (performance index *J*) can be considered the most important parameter in this assessment. This is because it is the real quantity "maximised" throughout the investigation and the main objective of every specific algorithm routine employed.

2. *Variation in the Performance index*: change in percentage between the initial value of the performance index *J* (with an initial "guess" profile) and the final measure obtained after the optimisation. In the case of no initial profile been used, the variation of the performance index has been fixed to 100%.

3. *Usefulness of the profile acquired*: a subjective parameter trying to go along with the industrial requirements in order for the temperature profile obtained to be implementable in practice. A smooth and even outline is required by the beer industry according to the literature and following the profile used at the time of the study.

4. *Time per iteration ratio*: the calculation of the total time required by a single iteration to be completed. Estimated by means of the total optimisation time divided by the total number of iterations.

5. *Computational cost*: the total time required by the optimisation algorithm to reached the maximum value of the performance index J. For this parameter to be legitimate the same PC has been used for all the algorithms.

Table 6.2 includes an abstract of the results obtained with each optimisation algorithm for every factor to be considered. With this, it includes in this order: the optimisation algorithm in question, the page number where the particular result can be found in this thesis, the maximum performance index value J obtained with the objective function (Factor 1); the percentage increase by the final J value obtained compared with the measure from the initial assumption (Factor 2); the convenience of the profile in practice according to the industrial requirements and with the help of Figures 6.1-6.6 presented earlier (Factor 3); the time in seconds required by one iteration to be completed (Factor 4) and at last the total time of the optimisation (Factor 5). It has to be mentioned that both of the last two factors can be compared for the different algorithms because the same personal computer have been used during the entire study.

Optimisation algorithm	**Page Number**	**Maximum J value**	**Variation in J (%)**	**Useful in practice?**	**Time per iteration^**	**Total time***
Golden Section Search	69	597.808	100.00	Yes	0.2744	0.0686
Gradient Method in Function Space	78	599.01	0.201	Maybe	0.816	0.6528
DISOPE	91	598.87	0.178	Maybe	0.6374	0.6905
GAs	124	600.13	100.00	Yes	177.603	444.007
SQP	163	600.204	29.476	Yes	4.386	2.047

^Time per iteration in seconds *Total time in minutes

Table 6.2 Overall contrast of the optimisation techniques

The value of the overall criteria can be considered an important assessment of the general performance of every algorithm. Herewith, conclusions can be made on the most suitable algorithm for the optimisation of the fermentation process modelled.

6.2 CONCLUSIONS AND FURTHER WORK

A simple approach in the selection of a helpful optimisation technique has been confirmed to be a valuable principle. The Golden Section Search by means of an optimal steady state solution (using MATLAB's *fminbnd* function) has achieved an excellent value and has been chosen as a good measure for an initial (starting point) temperature profile that can be used for the proposed algorithms.

Together with this, a robust nonlinear constrained optimisation technique such as Sequential Quadratic Programming have proven to perform well according to the particular factors discussed before. The flexibility of SQP to handle constraints is a decisive feature that can make the difference in the selection of this algorithm above the rest. Herewith, if any of these algorithms is to be applied to a different fermentation or industrial process in general, adaptability and convergence to the global optimum becomes an important issue. In this context, SQP provides excellent possibility for modification and also obtaining a maximum value of the objective function. The termination criterion for the cases considered is the variation in the objective function value from iteration to iteration; this because the number of iterations was set to 100 but was never reached, as included in the intial parameters used table in chapter 5. Furthermore, the implementation of the SQP algorithm with the help of the MATLAB Software can be performed by any control engineer with a reasonable good knowledge of the MATLAB programming language.

Genetic Algorithms have demonstrated to be very helpful for the optimisation of the mathematical model of the process, although the computational effort needed is always the main concern. In the context of this thesis, since all the optimisation work completed involve an "off-line" approach, the total time required can not necessarily be a great influential parameter. Together with this, the adaptability and robustness of this stochastic method in finding a global optimum for the objective function defined without the need of providing an initial profile, can be a crucial factor for a specific application with another process. For this algorithm, the termination criteria applied has been the total number of generations allowed. With this, different amount of generations have

been used for every different case in order to obtain a suitable optimised temperature profile.

The remaining two methods used, the Gradient Method in Function Space and the DISOPE algorithm, can be considered an intermediate point between a complex technique and a simplistic approach. By means of using the Golden Section Search method's optimised temperature profile as an initial profile, they are able to obtain a satisfactory optimal solution in a desirable time that translates in the computational cost considered to be minimum. Nonetheless, a disadvantage has been the prerequisite for an appropriate initial guess in order to perfom sufficiently well. The termination criteria for these methods, have been based on the actual improvement from one iteration to the other but also considering that the optimised profile still maintains a useful outline for practical application. With this, better results can be obtained in further work with these techniques by means of a smoothing technique applied after the optimisation, allowing the algorithm to work with more iterations.

It is important to note, that even though the termination criterion differs slighlty depending on the optimisation technique used, the best profile has been obtained among every case considered, thus giving a good representation of the capabilities of the algorithm in question.

In general, further work can be focus on the validation of these methods for the optimisation of other industrial processes. The techniques presented in this thesis can then be tested in different circumstances and a more general evaluation been made. Herewith, depending on the particular process to be optimised, different results may be obtained using these methods since they may performed better or worse for other process models.

New and/or revised optimisation methods can also be used for the optimisation of the fermentation process selected in this thesis; this in order to continue with the comparison under a similar situation. Neural Networks and Fuzzy Logic are only a few of these techniques since new developments towards optimal control have been developed recently and can be exploited. An introduction on Neural Networks for System

Identification of the selected beer process has been included in Appendix C as a starting point to pursue further research on this particular area.

6.3 LIST OF CONFERENCES AND PUBLICATIONS FROM THIS RESEARCH

Carrillo-Ureta, G.E.; Roberts, P.D. and Becerra, V.M. (2002a). Optimal Control of a Beer Fermentation Process: A Comparative Study. Submitted for the consideration of the *Control Engineering Practice Journal* in December, 2002.

Carrillo-Ureta, G.E.; Roberts, P.D. and Becerra, V.M. (2002b). Optimal Control of a Beer Fermentation. Submitted for the consideration of the *ECC 2003 Conference* to be held in September 2003 in Cambridge, U.K.

Carrillo-Ureta, G.E. (2002c). Optimisation of a Beer Fermentation Process Using Sequential Quadratic Programming. *Proceedings of the Postgraduate Symposium, UKACC Conference in Control 2002*. pp. 180-185. Sheffield, U.K.

Carrillo-Ureta, G.E.; Roberts, P.D. and Becerra, V.M. (2001a). Genetic Algorithms for Optimal Control of Beer Fermentation. *Proceedings of the 2001 IEEE International Symposium on Intelligent Control*. pp. 391-396. Mexico City, Mexico.

Carrillo-Ureta, G.E.; Roberts, P.D. and Becerra, V.M. (2001b). Simulación y Control Optimo de Procesos de Fermentación. Presented in the *1st Congress of the Panamanian Automatic Control Association (APCA)*. August 2001 in Panama City, Panama.

Carrillo-Ureta, G.E. (1999). *Optimal Control of Fermentation Processes*. M.Phil. to PhD Transfer Report CERC/LKC/172. City University, London.

APPENDIX A

THE GENETIC ALGORITHMS TOOLBOX FUNCTIONS DESCRIBED

The major elements of the genetic algorithm toolbox are described as follows (Chipperfield et al, 1994 and Chishimba, 1998):

A. Creating Populations: The GA Toolbox supports binary, integer and floating-point chromosome representations. Binary and integer populations may be initialised using the Toolbox function to create binary populations.

1. ***crtbase***: Builds a vector describing the integer representation used.
   E.g. To create a vector containing the first five elements in base 6 and the last three elements in base two:
   pop = crtbase ([5 3] , [6 2])
   pop = 6 6 6 6 6 2 2 2

2. ***crtbp***: Creates an initial population consisting of random chromosomes. E.g. To create a random population (matrix) of size 5 x 6 in base 4:
   pop = crtbp (5, 6, 4)

   pop = 3 3 2 1 0 0
   0 1 3 3 1 0
   2 0 3 3 3 2
   1 3 2 1 0 1
   3 1 0 3 0 0

3. ***crtrp***: Creates a real valued initial population consisting of random individuals.
   E.g. To create a real-valued random population of size 3 x 4 where the variables in the first, second and third columns are in the ranges of 1 to 5, -4 to 1 and 8 to 13 respectively:
   range = [1 -4 8 ; 5 1 13]
   pop = crtrp (4, range)
   pop = 1.0611 -1.6700 9.0132

3.9871 -1.9068 11.3607

2.7804 0.2311 12.1906

4.7273 -1.3742 8.0982

B. Fitness Assignment: Is normally used to transform the objective function value into a measure of relative fitness. The toolbox supports both linear and non-linear ranking methods and includes a simple linear scaling function for completeness.

1. ***ranking***: Ranks individuals according to their objective values and returns a column vector containing the corresponding individual fitness values. This function ranks individuals for minimisation.

   E.g. To evaluate the fitness with linear ranking, given the objective values of a population of 8 individuals:

   values1 = [3 ; 6 ; 7 ; 2 ; 8 ; 10 ; 1 ; 5]

   fitness = ranking (values1)

   fitness = 1.4286

   0.8571

   0.5714

   1.7143

   0.2857

   0

   2.0000

   1.1429

2. ***scaling***: Converts the objective values of a population into a fitness measure with a known upper bound, such that: $F(x_i) = af(x_i) + b$

   E.g. To convert the objective values of 6 individuals into a fitness measure with upper bound of 2:

   values2 = [1 ; 5 ; 3 ; 2 ; 8 ; 7]

   function = scaling (values2)

   delta = 3.6667

   a = 1.1818

   b = -0.7879

function = 0.3939
5.1212
2.7576
1.5758
8.6667
7.4848

C. Selection Functions:

1. ***reins***: Performs insertion of offspring into the current population, replacing parents with offspring and returning the resulting population.
   E.g. To produce a new population from a population of 7 parents and a population of 5 offspring:
   popul1 = [1 ; 3 ; 5 ; 7 ; 9 ; 11 ; 13]
   offspr = [21 ; 22 ; 23 ; 24 ; 25]
   newpop1 = reins (popul1, offspr)
   newpop1 = 21
   3
   22
   23
   25
   11
   24

2. ***rws***: Probabilistically selects individuals for reproduction according to their fitness in the current population using roulette wheel selection.
   E.g. To select the indices of 4 individuals considering a population of 7 individuals with the assigned fitness values:
   asgfit2 = [2.5 ; 0.4 ; 4.7 ; 1.1 ; 0.9 ; 3.2]
   newpop2 = rws (asgfit2 , 4)
   newpop2 = 1
   3
   6
   6

3. ***select***: Performs selection of individuals from a population and returns the selected individuals in a new population. Each row corresponds to one individual.

   E.g. To select 4 individuals by stochastic universal sampling (*sus*). considering a population of 5 individuals with the assigned fitness values:

   popul3 = [3 23 43 ; 5 25 45 ; 7 27 47 ; 9 29 49 ; 11 31 51]

   asgfit3 = [3.5 ; 2.7 ; 7.5 ; 6.4 ; 12.9]

   newpop3 = select ('sus' , popul3 , asgfit3)

   newpop3 = 11 31 51
   7 27 47
   9 29 49
   11 31 51
   3 23 43

4. ***sus***: Probabilistically selects individuals for reproduction according to their fitness in the current population using stochastic universal sampling.

   E.g. To select the indices of 4 individuals considering a population of 6 individuals with the assigned values:

   asgfit4 = [3.2 ; 1.6 ; 2.5 ; 1.3 ; 5.4 ; 8.6]

   newpop4 = sus (asgfit4 , 4)

   newpop4 = 3
   1
   6
   5

D. Mutation Operators:

1. ***mut***: Takes the representation of the current population and mutates each element with a given probability using a discrete mutation operator. It is a low-level mutation function normally called by *mutate*.

   E.g. To mutate with default probability (0.7/length of chromosome) a binary population with 5 individuals each of length 6:

   pop1 = [0 1 0 1 0 0 ; 1 1 1 0 0 0 ; 0 0 0 1 1 0 ; 1 0 0 1 0 1 ; 1 1 0 0 0 0]

newpop1 = mut (pop1)

newpop1 = 0 1 0 1 1 0
1 1 1 0 0 0
0 1 0 1 1 0
1 0 0 1 1 1
1 1 0 0 0 0

2. ***mutate***: Performs mutation of individuals from a population and returns the mutated individuals in a new population. Each row corresponds to one individual. This is a high-level mutation used in conjunction with *mutbga* and *mut*.

3. ***mutbga***: Takes the real-valued population, mutates each variable with given probability and returns the population after mutation. It is a low-level mutation function normally called by *mutate*.

   E.g. To mutate a population of 3 real-valued individuals with length 4, with certain bounds with a mutation probability 1/5 and no shrinking of the mutation range:

   pop3 = [35 86 50 22 ; 33 99 41 9 ; 23 101 65 -7]

   bound3 = [0 75 40 -20 ; 50 125 80 40]

   newpop3 = mutbga (pop3 , bound3 , [1/5 1.0])

   newpop3 = 35.0000 86.0000 50.0000 22.0000
   33.0000 99.0000 41.6251 9.9377
   23.0000 94.7500 65.0000 -7.0000

E. Crossover Operators:

1. ***recdis***: Performs discrete recombination between pairs of individuals in the current population and returns a new population after mating. It is a low-level recombination function normally called by *recombin*.

   E.g. To perform discrete recombination considering a population with 4 real-value individuals of length 3:

   inpop1 = [83 48 20 ; 63 31 -9 ; 52 -15 -37 ; 102 65 -7]

   nwpop1 = recdis (inpop1)

```
nwpop1 =     83   31   -9
             63   31   20
            102  -15   -7
            102   65   -7
```

2. ***recint***: Performs intermediate recombination between pairs of individuals in the current population and returns a new population after mating. It is a low-level recombination function normally called by *recombin*.

   E.g. To perform intermediate recombination considering a population with 4 real-value individuals of length 3:

```
inpop2 = [ 53 28 12 ; 37 15 -28 ; -23 51 -7 ; 42 58 -22 ]
nwpop2 = recint ( inpop2 )
nwpop2 =     39.1685   12.9240   -6.1987
             55.6035    22.3633  -29.3118
             24.2061   57.9944  -17.4220
             13.6133   58.3124  -13.8758
```

3. ***reclin***: Performs line recombination between pairs of individuals in the current population and returns a new population after mating. It is a low-level recombination function normally called by *recombin*.

   E.g. To perform line recombination considering a population with 3 real-value individuals of length 4:

```
inpop3 = [ 34 -25 18 -66 ; 38 -16 -4 8 ; -12 24 52 -23 ]
nwpop3 = reclin ( inpop3 )
nwpop3 =     33.9052  -25.2132   18.5212  -67.7531
             37.1874  -17.8284    0.4694   -7.0333
            -12.0000   24.0000   52.0000  -23.0000
```

4. ***recmut***: Performs line recombination with mutation features between pairs of individuals in the current population and returns a new population after mating. It is a low-level recombination function normally called by *mutate*.

   E.g. To perform line recombination with mutation features considering a population with 3 real-valued individuals of length 3:

```
inpop4 = [ 41 50 -28 ; 33 64 -8 ; 29 55 3 ]
```

```
bound4 = [ 10 25 -40 ; 50 100 20 ]
nwpop4 = recmut ( inpop4 , bound4 )
nwpop4 =      41.1573   49.4840  -27.4103
              32.8427   64.5160   -8.5897
              29.0000   55.0000    3.0000
```

5. ***recombin***: Performs recombination of individuals from a population and returns the recombined individuals in a new population. This high-level function checks the consistency of the input parameters and calls the low-level recombination function.

6. ***xovdp***: Performs double-point crossover between pairs of individuals contained in the current population according to the crossover probability and returns a new population after mating. It is a low-level crossover function normally called by *recombin*.

   E.g. To perform double-point crossover considering a population with 4 real-valued individuals of length 3 and a crossover rate of 0.9:

```
inpop6 = [ 13 9 -18 ; 31 23 -15 ; 19 35 11 ; 5 67 32 ]
nwpop6 = xovdp ( inpop6 , 0.9 )
nwpop6 =      13    9   -15
              31   23   -18
              19   67    11
               5   35    32
```

7. ***xovdprs***: Performs double-point reduced surrogate crossover between pairs of individuals contained in the current population according to the crossover probability and returns a new population after mating. It is a low-level crossover function normally called by *recombin*.

   E.g. To perform double-point reduced surrogate crossover considering a population with 4 binary individuals of length 4 and a crossover rate of 0.7:

```
inpop7 = [ 1 0 1 1 ; 1 1 1 0 ; 0 1 1 0 ; 0 0 0 0 ]
nwpop7 = xovdprs ( inpop7 )
nwpop7 =      1    0    1    0
              1    1    1    1
```

0 1 0 0

0 0 1 0

8. ***xovmp***: Performs multi-point crossover between pairs of individuals in the current population and returns a new population after mating. Number of cross-points and the use of reduced surrogate can be specified. It is a low-level crossover function normally called by all other crossover functions.

   E.g. To perform single-point crossover with no reduced surrogate considering a population with 6 real-valued individuals of length 3 and a crossover rate of 0.7:

   inpop8 = [32 19 83 ; 15 38 -51 ; 90 50 16 ; 9 73 23 ; 13 21 66 ; 11 25 88]

   nwpop8 = xovmp (inpop8 , 0.7 , 1 , 0)

   nwpop8 =

32	38	-51
15	19	83
90	50	23
9	73	16
13	21	88
11	25	66

9. ***xovsh***: Performs shuffle crossover pairs of individuals contained in the current population according to the crossover probability and returns a new population after mating. It is a low-level crossover function normally called by *recombin.*

   E.g. To perform shuffle crossover considering a population with 6 binary individuals of length 4 and a crossover rate of 0.4:

   inpop9 = [1 0 1 1 ; 1 1 1 0 ; 0 1 1 0 ; 0 0 0 0 ; 1 1 0 0 ; 1 1 1 1]

   nwpop9 = xovsh (inpop9 , 0.4)

   nwpop9 =

1	1	1	1
1	0	1	0
0	0	0	0
0	1	1	0
1	1	0	0
1	1	1	1

10. ***xovshrs***: Performs shuffle crossover with reduced surrogates between pairs of individuals contained in the current population according to the crossover probability and returns a new population after mating. It is a low-level crossover function normally called by *recombin*.

E.g. To perform shuffle crossover with reduced surrogates considering a population with 5 real-valued individuals of length 3 and a crossover rate of 0.8:

inpop10 = [22 90 32 ; 6 9 62 ; 12 34 28 ; 31 11 67 : 19 51 87]

nwpop10 = xovshrs (inpop10 , 0.8)

nwpop10 =

6	9	32
22	90	62
31	11	28
12	34	67
19	51	87

11. ***xovsp***: Performs single-point crossover between pairs of individuals contained in the current population according to the crossover probability and returns a new population after mating. It is a low-level crossover function normally called by *recombin*.

E.g. To perform single-point crossover considering a population with 6 binary individuals of length 4 and a crossover rate of 0.6:

inpop11 = [0 0 0 1 ; 1 0 1 1 ; 0 0 1 0 ; 0 0 0 1 ; 1 0 0 1 ; 1 1 1 0]

nwpop11 = xovsp (inpop11 , 0.6)

nwpop11 =

0	0	0	1
1	0	1	1
0	0	1	0
0	0	0	1
1	0	1	0
1	1	0	1

12. ***xovsprs***: Performs single-point reduced surrogate crossover between pairs of individuals contained in the current population according to the crossover probability and returns a new population after mating. It is a low-level crossover function normally called by *recombin*.

E.g. To perform shuffle crossover with reduced surrogates considering a population with 6 real-valued individuals of length 3 and a crossover rate of 0.5:

```
inpop12 = [ 21 4 13 ;  16 99 25 ; 19 44 81 : 12 18 72 : 92 34 80 : 7 -9 15 ]
nwpop12 = xovsprs ( inpop12 , 0.5 )
nwpop12 =    21   99   25
             16    4   13
             19   44   81
             12   18   72
             92   34   80
              7   -9   15
```

F. Subpopulation Support:

1. ***migrate***: Performs migration of individuals between subpopulations in the current population and returns the population after migration. The number of subpopulations, rate of migration, the migration selection method and the structure of the subpopulations for migration can be indicated.

   E.g. To perform migration of 20% of the individuals of one subpopulation and replace them with uniformly chosen individuals from all other subpopulations:

```
popul = [ 13 9 -18 ; 31 23 -15 ; 19 35 11 ; 5 67 32 ; 3 45 50 ; 88 10 22]
newpop = migrate ( popul , 6 )
newpop =    88   10   22
            19   35   11
            31   23  -15
            88   10   22
            13    9  -18
             3   45   50
```

G. Utility Functions:

1. ***bs2rv***: Decodes the binary representation of the population into vectors of real values. The chromosomes are seen as concatenated strings of given length.

and decoded into real numbers over a specified interval using either standard binary or Gray coding.

E.g. To convert the Gray code binary representation created using the *crtbp* function representing a set of single decision variables in the range [-3 . 20]. to real-valued phenotypes using arithmetic scaling:

```
popul = crtbp (6, 8 )
popul =     1   1   0   1   1   1   1   1
            0   1   0   0   1   0   1   1
            0   0   0   1   0   1   0   1
            0   0   1   1   1   1   0   0
            1   1   1   0   1   0   1   0
            1   1   1   1   0   0   1   1
repres = [ 8 ; -3 ; 20 ; 1 ; 0 ; 1 ; 1 ]
phenot = bs2rv (popul , repres )
phenot =    10.4392
             7.2824
            -0.7451
             0.6078
            13.1451
            11.6118
```

2. ***rep***: Performs replication of a matrix specified and returns the replicated matrix. It is a low-level replication function normally called by a number of functions.

   E.g. To perform replication of a given matrix size 3 x 4 including 2 vertical and 3 horizontal replications:

```
inimat = [ 2 4 6 8 ;  9 7 5 3 ; -1 -3 -5 -7 ]
newmat = rep (inimat , [ 2 3 ] )
newmat =    2   4   6   8   2   4   6   8   2   4   6   8
            9   7   5   3   9   7   5   3   9   7   5   3
           -1  -3  -5  -7  -1  -3  -5  -7  -1  -3  -5  -7
            2   4   6   8   2   4   6   8   2   4   6   8
            9   7   5   3   9   7   5   3   9   7   5   3
           -1  -3  -5  -7  -1  -3  -5  -7  -1  -3  -5  -7
```

The main data structures included in the Genetic Algorithm Toolbox are as follow (Chipperfield, 1994):

- Chromosomes: Stores an entire population in a single matrix with the number of individuals in the population in the column vector and the length of the genotypic representation of these individuals in the row vector. This representation does not force a structure on the chromosome structure; the only requirement is that all chromosomes are of equal length.

$$Chromosome=\begin{bmatrix} g_{1,1} & g_{1,2} & \cdot & g_{1,Lind} \\ g_{2,1} & g_{2,2} & \cdot & g_{2,Lind} \\ \cdot & \cdot & \cdot & \cdot \\ g_{Nind,1} & g_{Nind,2} & \cdot & g_{Nind,Lind} \end{bmatrix} \begin{matrix} \text{individual 1} \\ \text{individual 2} \\ \\ \text{individual } N \end{matrix}$$

- Phenotypes: The also called decision variables are obtained by applying some mapping from the chromosome representation into the decision variable space. Each string contained in the chromosome structure decodes to a row vector (*Nvar*) according to the number of dimensions in the search space and corresponding to the decision variable vector value.

$$Phenothype=\begin{bmatrix} x_{1,1} & x_{1,2} & \cdot & x_{1,N\,var} \\ x_{2,1} & x_{2,2} & \cdot & x_{2,N\,var} \\ \cdot & \cdot & \cdot & \cdot \\ x_{Nind,1} & x_{Nind,2} & \cdot & x_{Nind,N\,var} \end{bmatrix} \begin{matrix} \text{individual 1} \\ \text{individual 2} \\ \\ \text{individual } N \end{matrix}$$

- Objective Function Values: An objective function is used to evaluate performance of the phenotypes in the problem domain. Values are stored in a numerical matrix where *Nobj* as the number of objectives.

$$Objective=\begin{bmatrix} y_{1,1} & y_{1,2} & \cdot & y_{1,N\,var} \\ y_{2,1} & y_{2,2} & \cdot & y_{2,N\,var} \\ \cdot & \cdot & \cdot & \cdot \\ y_{Nind,1} & y_{Nind,2} & \cdot & y_{Nind,N\,var} \end{bmatrix} \begin{matrix} \text{individual 1} \\ \text{individual 2} \\ \\ \text{individual } N \end{matrix}$$

- Fitness Values: Are derived from the objective function values through a scaling or ranking function. Fitnesses are non-negative scalars and are stored in column vector. For multiobjective functions, the fitness of a particular individual is a function of a vector of objective function values.

$$Fitness = \begin{bmatrix} f_1 \\ f_2 \\ \cdot \\ f_{Nind} \end{bmatrix} \quad \begin{matrix} \text{individual 1} \\ \text{individual 2} \\ \\ \text{individual } N \end{matrix}$$

The GA Toolbox provides support for multiple populations using high-level genetic operator functions and a routine for exchanging individuals between subpopulations. The use of a single population divided into a number of subpopulations by modifying the use of data structures such that subpopulations are stored in contiguous blocks within a single matrix.

$$Chromosome = \begin{bmatrix} Ind_1 Subpop_1 \\ Ind_2 Subpop_1 \\ \cdot \\ Ind_N Subpop_1 \\ Ind_1 Subpop_2 \\ Ind_2 Subpop_2 \\ \cdot \\ Ind_N Subpop_2 \\ \cdot \\ Ind_1 Subpop_{SUBPOP} \\ Ind_2 Subpop_{SUBPOP} \\ \cdot \\ Ind_N Subpop_{SUBPOP} \end{bmatrix}$$

To allow the routines to operate independently on subpopulations, a number of high-level functions are provided that accept an optional argument that determines the number of subpopulations contained in a data structure.

APPENDIX B

SYSTEM IDENTIFICATION USING MATLAB

In order to perform the system identification, the input and output vectors of the process model are required. With the help of the SIMULINK model previously created and tested in Chapter 2, and by means of adding some white noise to the initial profile, an output of the process can be achieved. This model can be seen in Figure B.1.

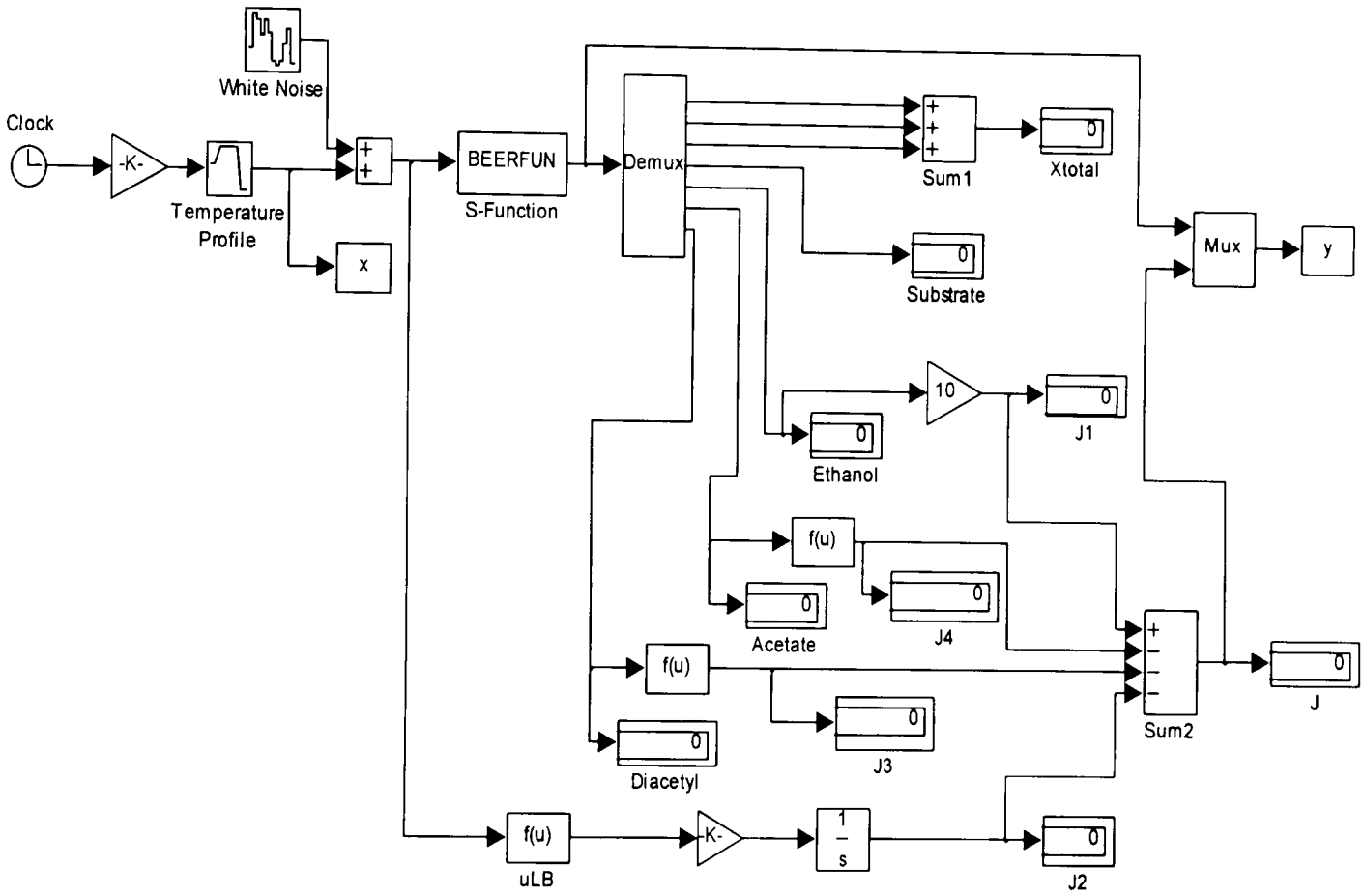

Figure B.1 SIMULINK Model used for System Identification

In this SIMULINK model, X and Y are the input and output vectors respectively. They are used as data in the System Identification toolbox in order to get the state space model matrices with the help of the System Identification Toolbox from the MATLAB Software (Ljung, 1995). System Identification is the art and science of building mathematical models from measured input-output data. Tasks such as examining the measured data, and creating new data sets from the original one by various pre-processing are required. Estimation of models using the data follows. The models are estimated within certain classes of candidate descriptions (model structures), typically by choosing that model that gives the best output fit to the

measured data. Quite a few models are normally estimated, and their properties are scrutinised and analysed in different model views.

ident is the graphical user interface to the System Identification Toolbox that helps with these tasks (Coleman et al, 1999). The ***ident*** window conducts all data handling, model estimation, and model analysis and also keeps a detailed record of all data sets and models. The process of system identification using ***ident*** involves the following six steps:

Importing data into IDENT: In order to create a data set from input and output signals available in the workspace (x and y), Import (from the Data pop-up menu) is selected. Earlier created models in the Toolbox's THETA-format can be imported from the MATLAB workspace using the Models pop-up menu. If the model originally was exported from ***ident***, it will be imported together with an *_info variable that contains relevant information.

Examining data: There are two views to examine data sets: double clicking the corresponding check boxes in the ***ident*** window opens these view windows. Time Plot shows the time plots of inputs and outputs of the selected data sets and Data Spectra gives the spectra (periodograms or estimated spactra) of inputs and outputs of the selected data sets.

Pre-processing data: The pop-up menu Pre-process provides several options for creating new data sets by modifying the Working Data set. The new data sets are inserted into the Data board. Select channels option allows selecting a subset of the input and output channels of the data set; the numbering of the inputs and outputs will be consistent throughout all data sets created from an original set. The Select range option allows selecting portions of the data set to be used for estimation and validation purposes. Remove means removes the mean values from both the input and output sequences. Remove trends estimates and removes a linear trend from the input and output signals. Filter pre-filters the data set. Resample allows to change the sampling interval, by interpolation and decimation. Finally, Quick start creates three new data sets from the Working Data set: First the mean values are removed,

then the new data is split into two halves, the first half becomes the new Working Data set, and the second half becomes the new Validation Data set.

Estimating models based on data: The pop-up menu Estimate provides various methods for estimating models based on the Working Data set. The models will be inserted into the Models board. Parametric models that opens a dialog, which allows to generate dynamic linear models with different structures, orders, and delays. A model structure characterises the relationship between input and output data and between unknown noise sources and the output data. Supported model structures include ARX, ARMAX, Output Error (OE), Box-Jenkins (BJ), State-Space, and others. Spectral model which, directly estimates the system's frequency response from the data using either Fourier Transform techniques or the Blackman-Tukey approach. The Correlation model directly estimates the system's transient response (impulse response) by correlating filtered versions of the input-output data. And the Quick start that performs correlation analysis and spectral analysis; computes a default ARX model with a delay heuristically determined based on the estimated impulse response of the system.

Analysing the models: The choice of text information and access to the model parameters or visual information by means of the six views to analyse and examine model properties. Checking the corresponding check boxes in the ***ident*** window opens these view windows. Model Output shows the simulated or predicted output of the selected models together with the actually measured output. Model Residuals shows the result of residual analysis for the each of the selected models. The residuals (prediction errors) are computed using the validation data set, and then their correlation functions are also shown. Transient Response presents the transient response (impulse or step response) of the selected models. Frequency Response shows the frequency functions of the selected models. Zeros and Poles demonstrates the Zeros and Poles of the selected models. Noise Spectrum shows the spectrum of the additive noise of the selected models. LTI Viewer invokes the Control System Toolbox's LTI Viewer.

Exporting resulting models for further use: The data sets and models created by IDENT are normally not MATLAB workspace variables. To further work with them

in command line mode, exporting them to the workspace by dragging and dropping their icon to the To Workspace icon in the ***ident*** window is necessary. Models are coded into the THETA format; THETA is a packed matrix containing information about both a model structure and its nominal or estimated parameters. it also contains other relevant information about the identification result. This model format is the basic format with the System Identification Toolbox. It is used by all parametric identification methods and it can be transformed to many other model representations.

After introducing the initial data from the identification SIMULINK model to the toolbox, a "quick start" is selected in the operations box in order to pre-process the information. The estimation of the model is accomplished choosing an ARX parametric model of order one (common poles, zeros and delay are one). Figures B.2 to B.9 show a comparison of the different state variable's profiles for the identification model pursued.

After the identification of the system the step responses (Figures B.10 to B.17) show how well the convergence of the parameters of the model is achieved. In all the Figures the blue line represents the Input Signal to be identified (original signal) and the red and light blue represents the Output Identified Signal.

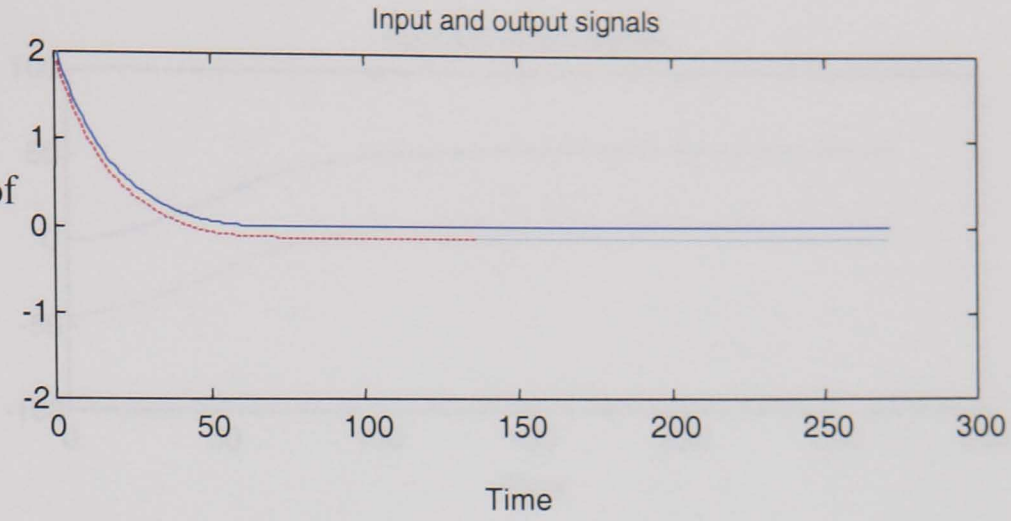

Figure B.2 Data Plot of Latent Biomass

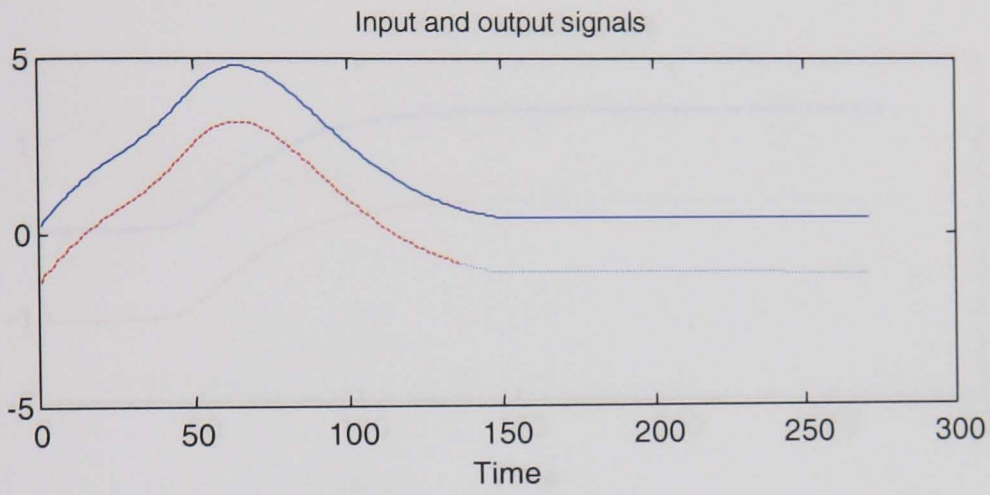

Figure B.3 Data Plot of Active Biomass

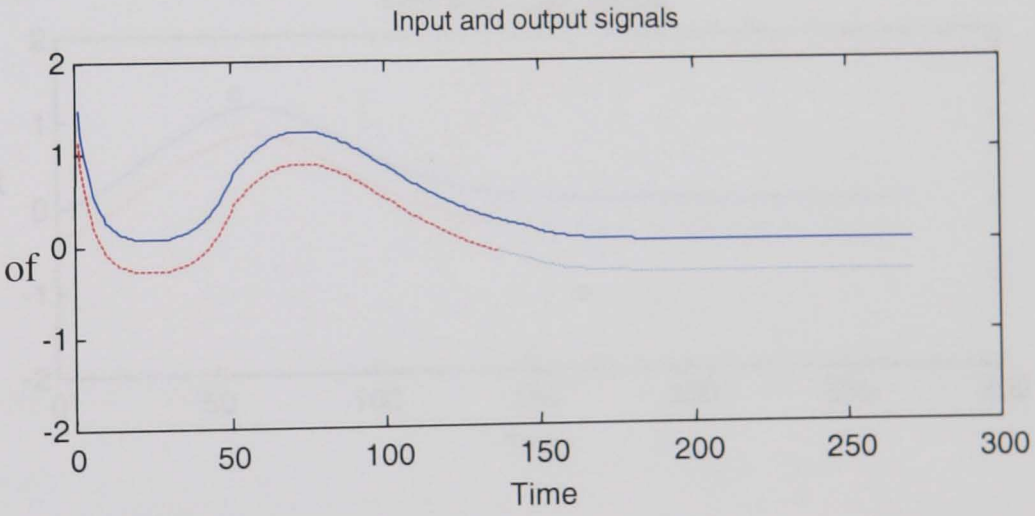

Figure B.4 Data Plot of Dead Biomass

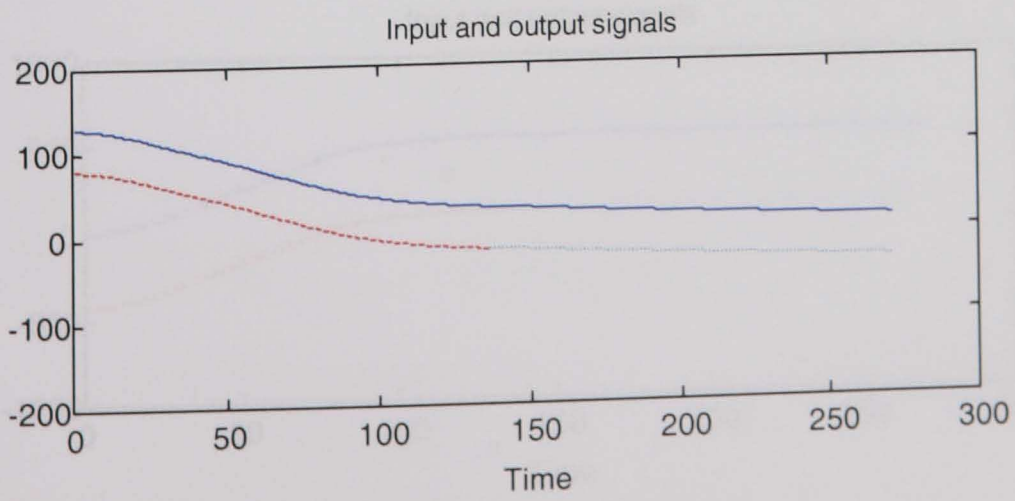

Figure B.5 Data Plot of Sugar Concentration

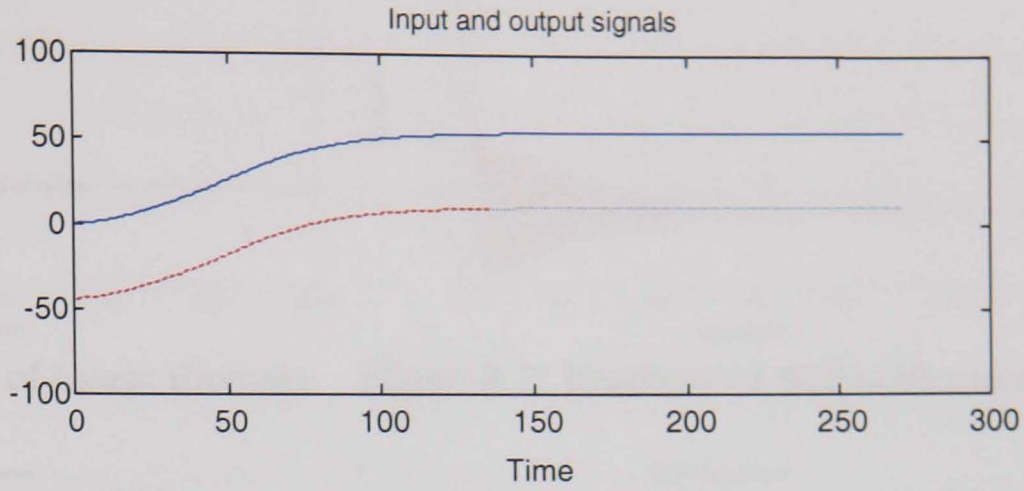

Figure B.6 Data Plot of Ethanol Concentration

Figure B.7 Data Plot of Ethyl Acetate Concentration

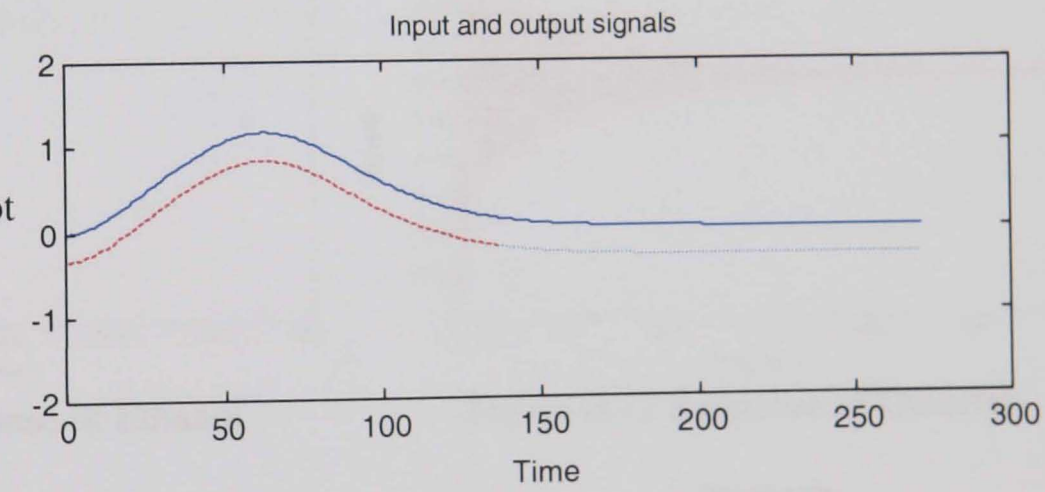

Figure B.8 Data Plot of Diacetyl Concentration

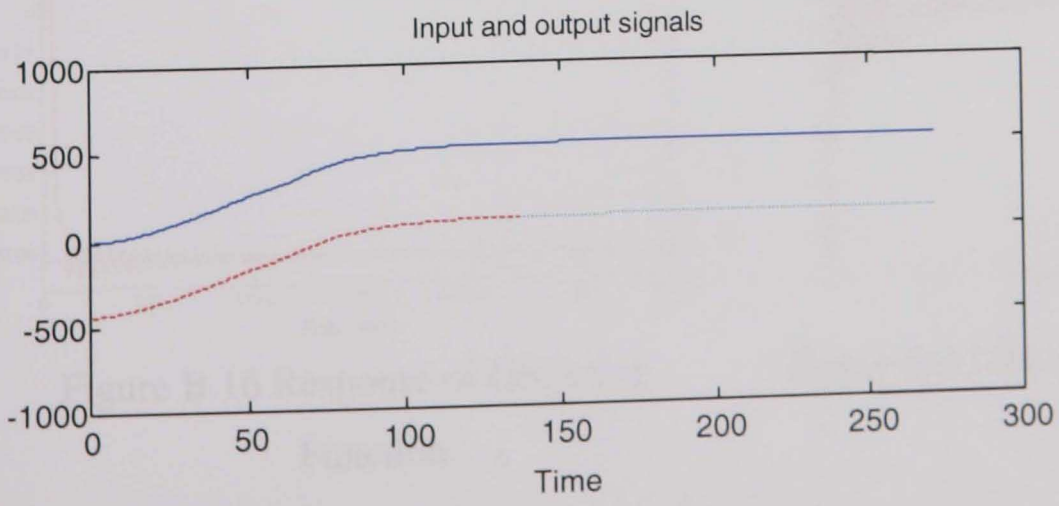

Figure B.9 Data Plot for the Objective Function

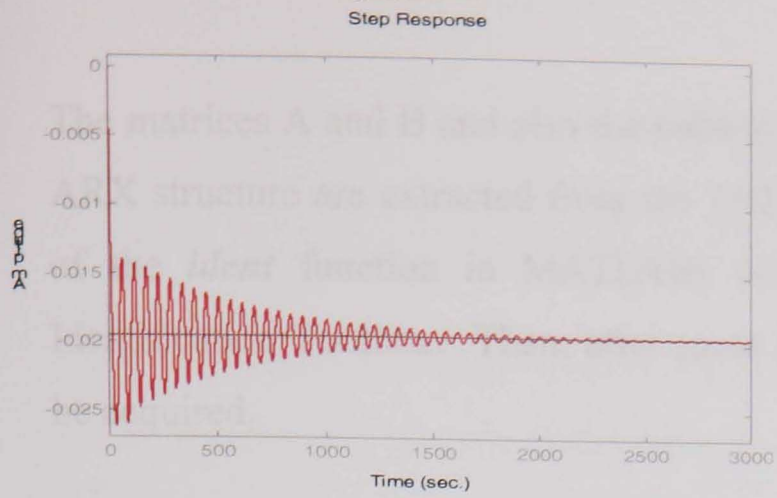

Figure B.10 Response of Latent Biomass

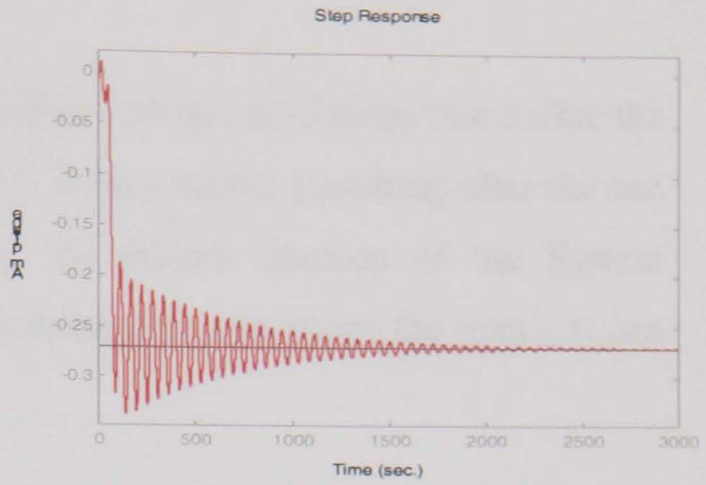

Figure B.11 Response of Active Biomass

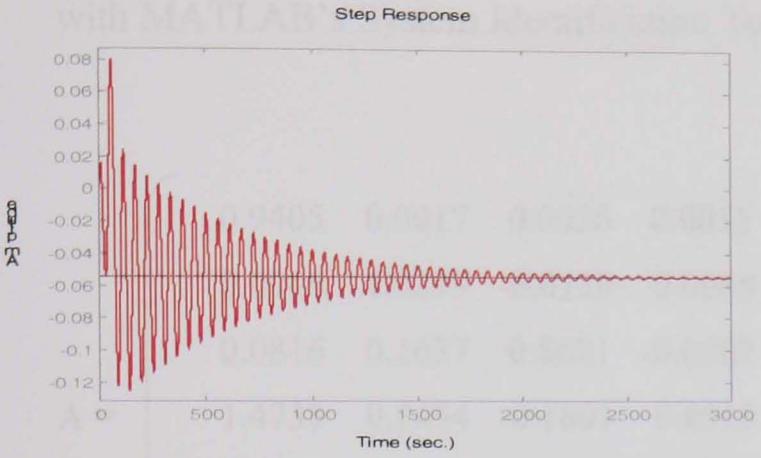

Figure B.12 Response of Dead Biomass

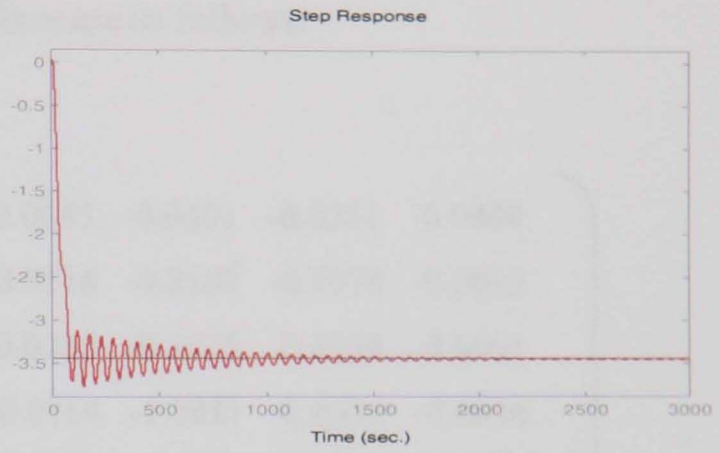

Figure B.13 Response of Sugar

Figure B.14 Response of Ethanol

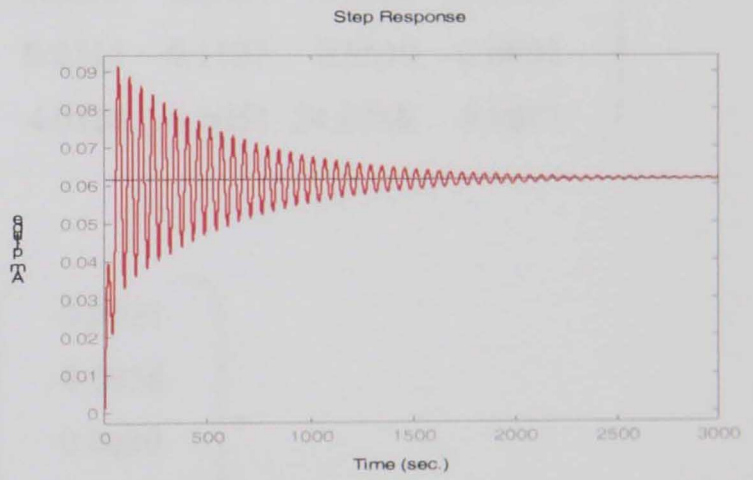

Figure B.15 Response of Diacetyl

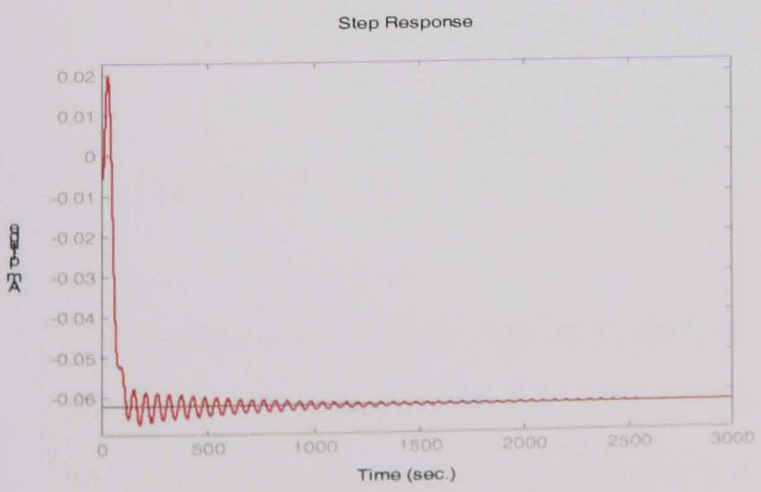

Figure B.16 Response of Objective Function

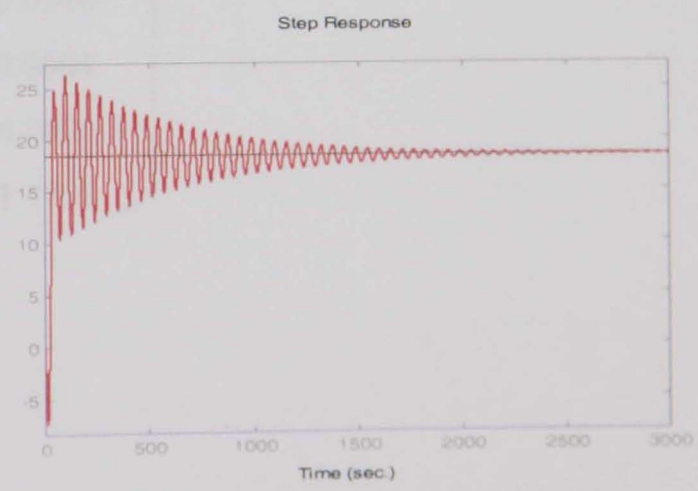

Figure B.17 Response of Temperature Profile

The matrices A and B and also the corresponding standard deviations that define the ARX structure are extracted from the THETA format model (resulting after the use of the ***ident*** function in MATLAB) using the ***th2arx*** function of the System Identification Toolbox. Then, after some mathematical operations the matrices can be acquired.

The state space model matrices (A and B) obtained from the system identification with MATLAB's System Identification Toolbox are as follows:

$$\mathbf{A} = \begin{pmatrix} 0.9405 & 0.0017 & 0.0026 & 0.0015 & 0.0043 & -0.0401 & -0.0351 & -0.0000 \\ 0.0946 & 1.0299 & -0.0130 & 0.0088 & 0.0516 & -0.8107 & -0.3978 & -0.0013 \\ 0.0816 & 0.1637 & 0.8621 & -0.0002 & 0.0155 & -0.3975 & -0.5354 & -0.0001 \\ 1.4733 & 0.7454 & -0.1807 & 0.8915 & -0.0764 & -1.7931 & -2.0919 & -0.0026 \\ -0.4569 & -0.0665 & 0.0563 & 0.0360 & 1.0972 & -0.8743 & -0.1763 & -0.0027 \\ -0.0048 & 0.0266 & 0.0182 & 0.0015 & 0.0099 & 0.8451 & -0.1536 & -0.0003 \\ -0.0396 & -0.0118 & 0.0056 & 0.0033 & 0.0111 & -0.1127 & 0.9510 & -0.0003 \\ -1.0513 & -10.3621 & -6.7555 & 2.0997 & 4.0186 & -6.0051 & 24.8758 & 0.9877 \end{pmatrix}$$

$$\mathbf{B} = \begin{pmatrix} -0.0025 \\ -0.0056 \\ 0.0030 \\ 0.0150 \\ 0.0146 \\ 0.0004 \\ -0.0026 \\ -1.1793 \end{pmatrix}$$

APPENDIX C

SYSTEM IDENTIFICATION WITH NEURAL NETWORKS

Using the Neural Networks System Identification Software by Norgaard et al. 2000; a new model representation of the process can be achieved.

The procedure used to identify a model of a dynamic system is represented in Figure C.1. The different stages of the diagram are briefly discussed below.

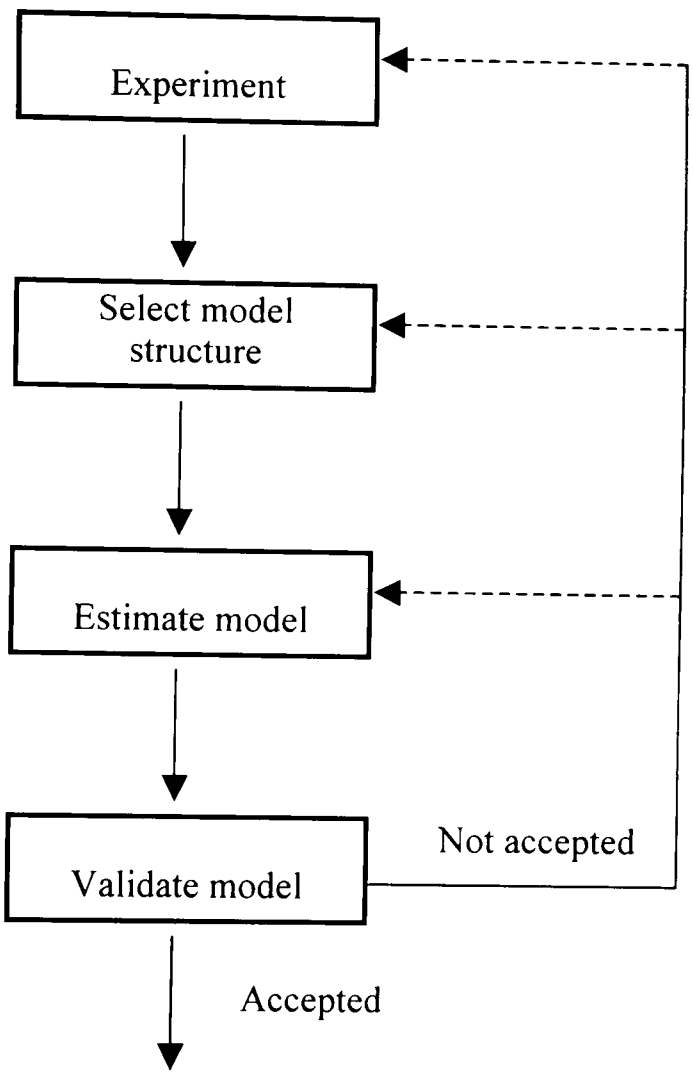

Figure C.1 The basic system identification procedure (Norgaard et al. 2000).

The experiment stage's idea is essentially collecting a set of data that describes how the system behaves during the whole process. It is extremely important, because of the non-linear black-box modelling considered, to collect a set of data that describes how the system behaves over its entire range of operation.

For the experiment, SIMULINK models of the beer process have been created with two different designs of input signals in order to observe the effect they have on the output. Input signals consisting of level changes at random instances have been used. These models are shown in figures C.2 and C.3.

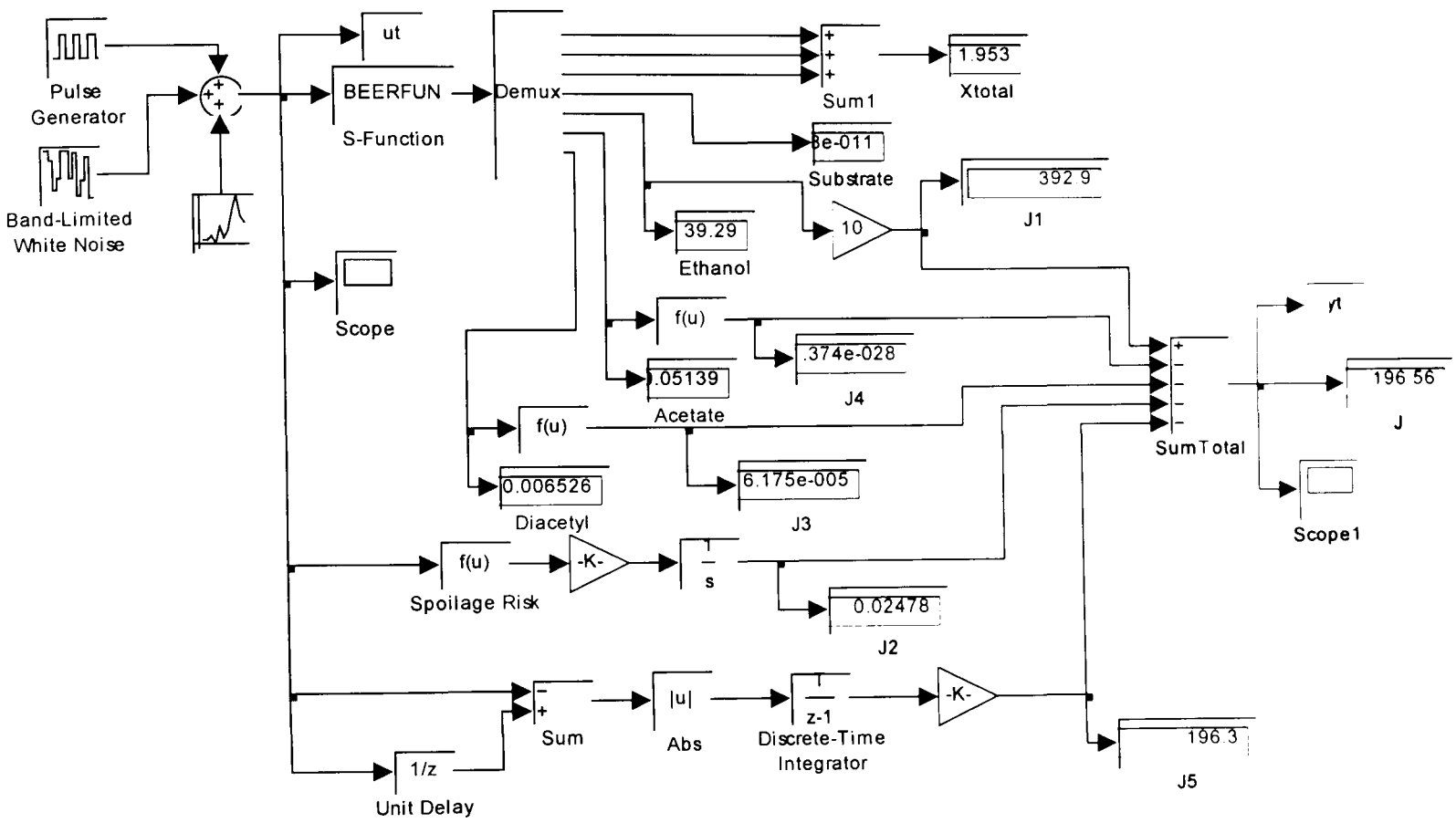

Figure C.2 SIMULINK model with input data "ut", resulting in output data "yt"

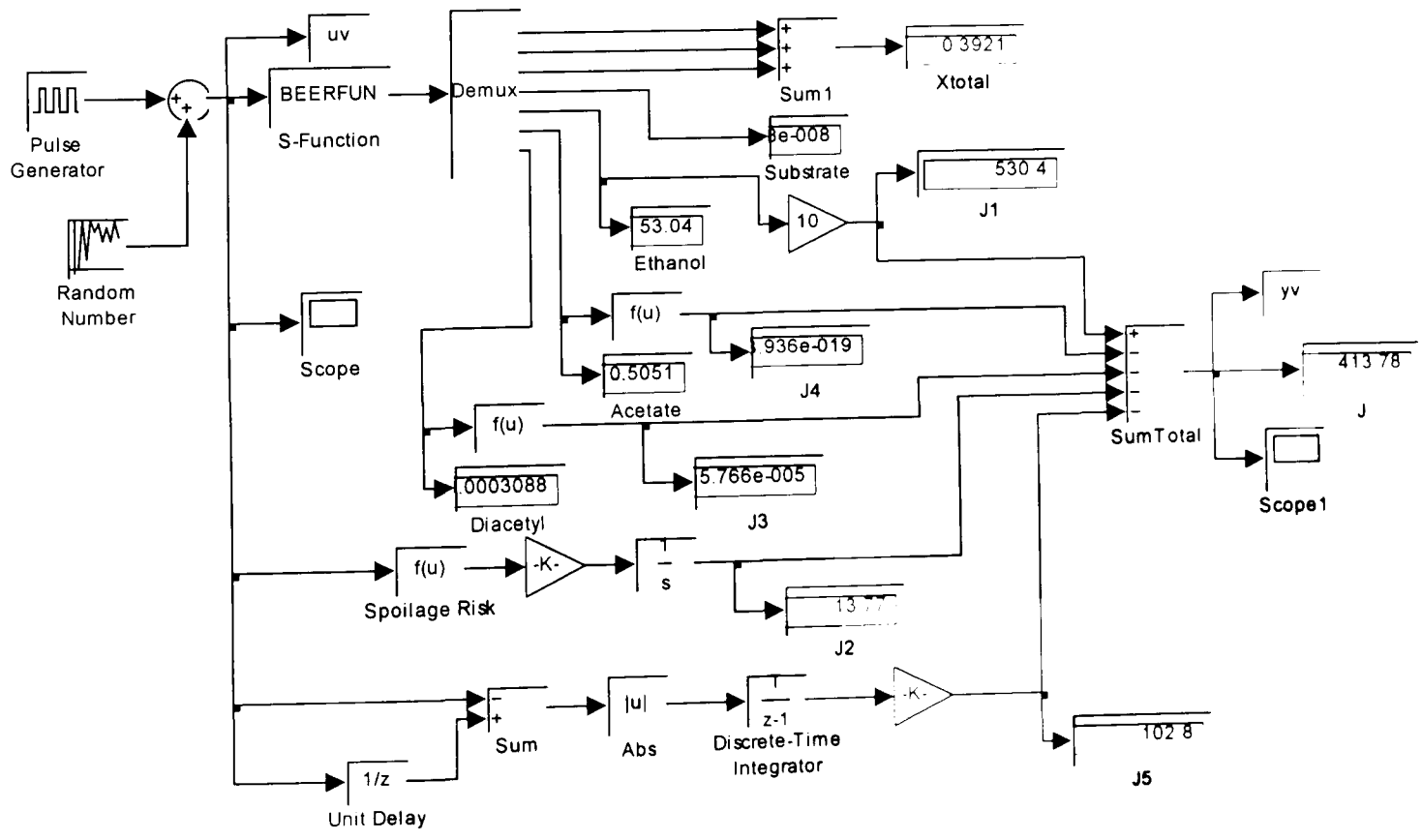

Figure C.3 SIMULINK model with input data "uv", resulting in output data "yv"

With these two different signals applied to the beer process and its corresponding outputs, deducing a model of the system can be accomplished. The first sets of input/outputs have been included in figures C.4 and C.5 (°C vs sample number and *J* value vs sample number).

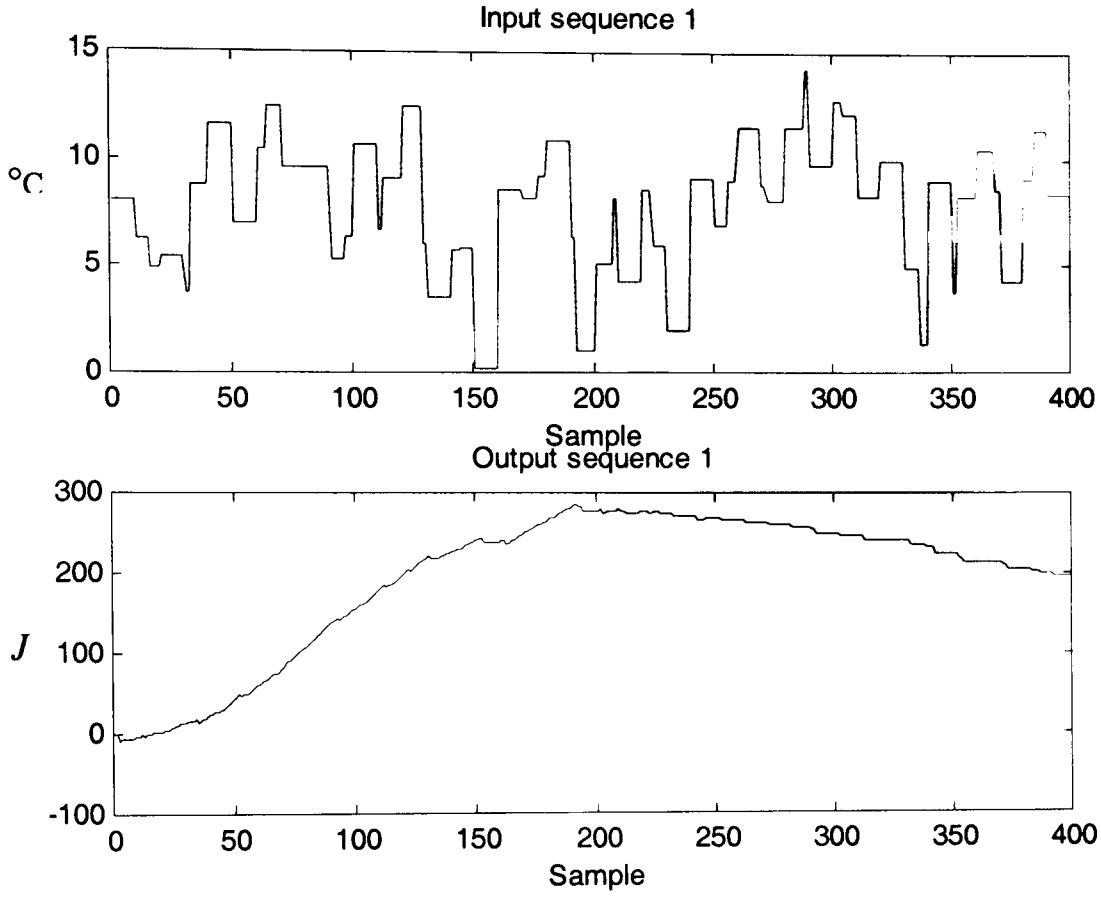

Figure C.4 Graphs representing Input/Output Sequence 1 (ut and yt)

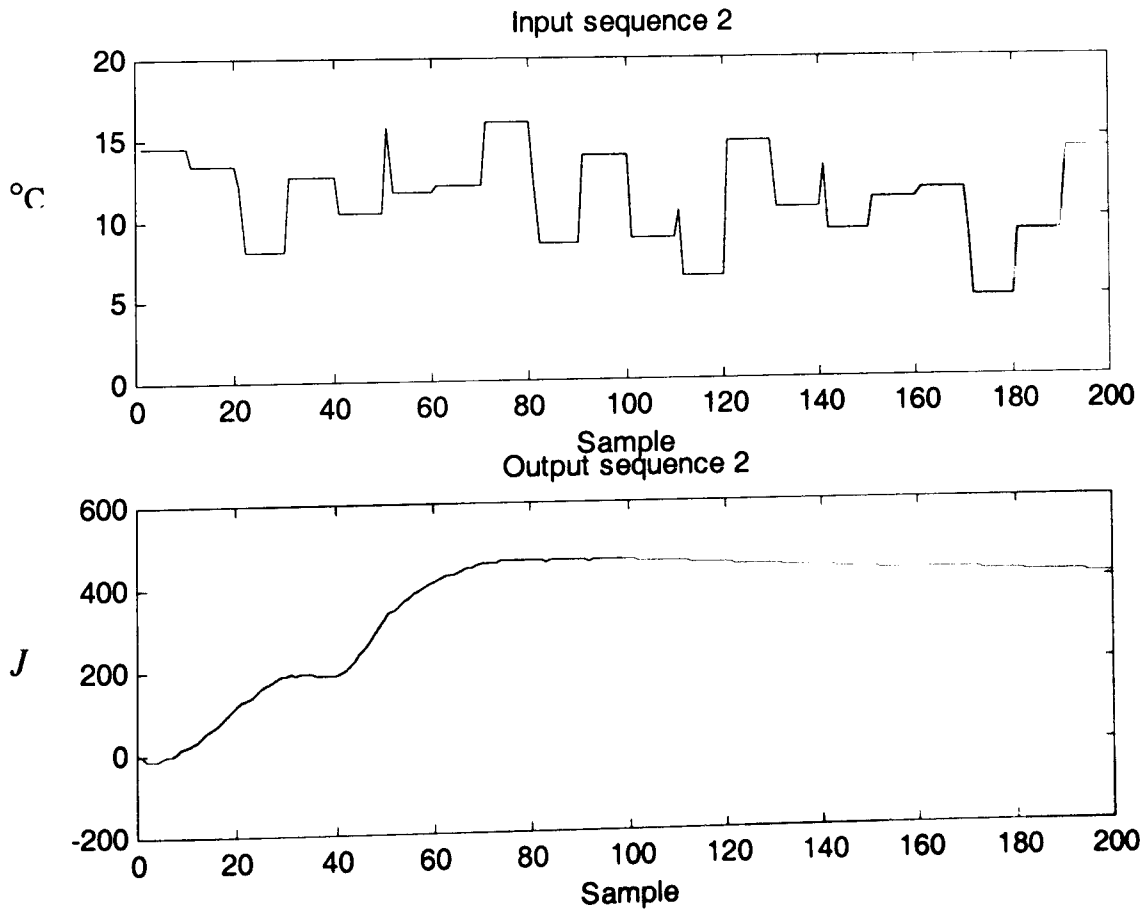

Figure C.5 Graphs representing Input/Output Sequence 2 (uv and yv)

For the second stage, selecting a model structure implies choosing a set of regressors (inputs) specifying how to combine them into a one-step ahead prediction. It is not possible to determine the optimal model structure, so instead a strategy, reasonably effortless, should be chosen that finds a structure that is sufficiently close to optimal. The NNARX (Neural Network AutoRegressive, eXternal input) model structure, the first choice among the structures, has the advantage that none of the regressors depend on past outputs of the model, which ensures that the predictor remains stable. This not only facilitates training, but also in general will result in more robust models.

Since the knowledge about the system is limited, the method based on Lipschitz quotients is considered. A wrong choice of lag space may have a unsuccessful impact on some control applications, so it is necessary to determine both a sufficiently large lag space and an adequate number of hidden units (Norgaard et al, 2000). With this, figure C.6 illustrates the method applied to the data set obtained from the simulated beer experiment. The number of past inputs and outputs are increase simultaneously from n=m=1 to n=m=7.

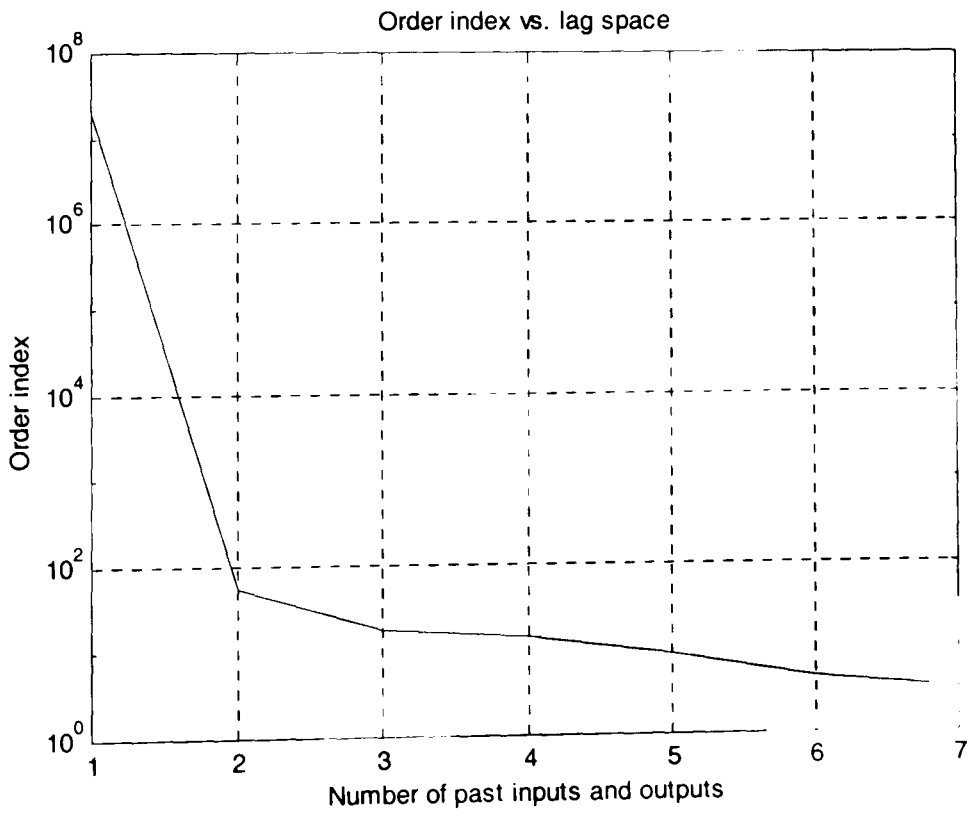

Figure C.6 The order index criterion evaluated for different lag spaces

Figure C.7 shows the lag space recommended for selection in red in a 3D graph using the "knee-point" of the curve as a reference.

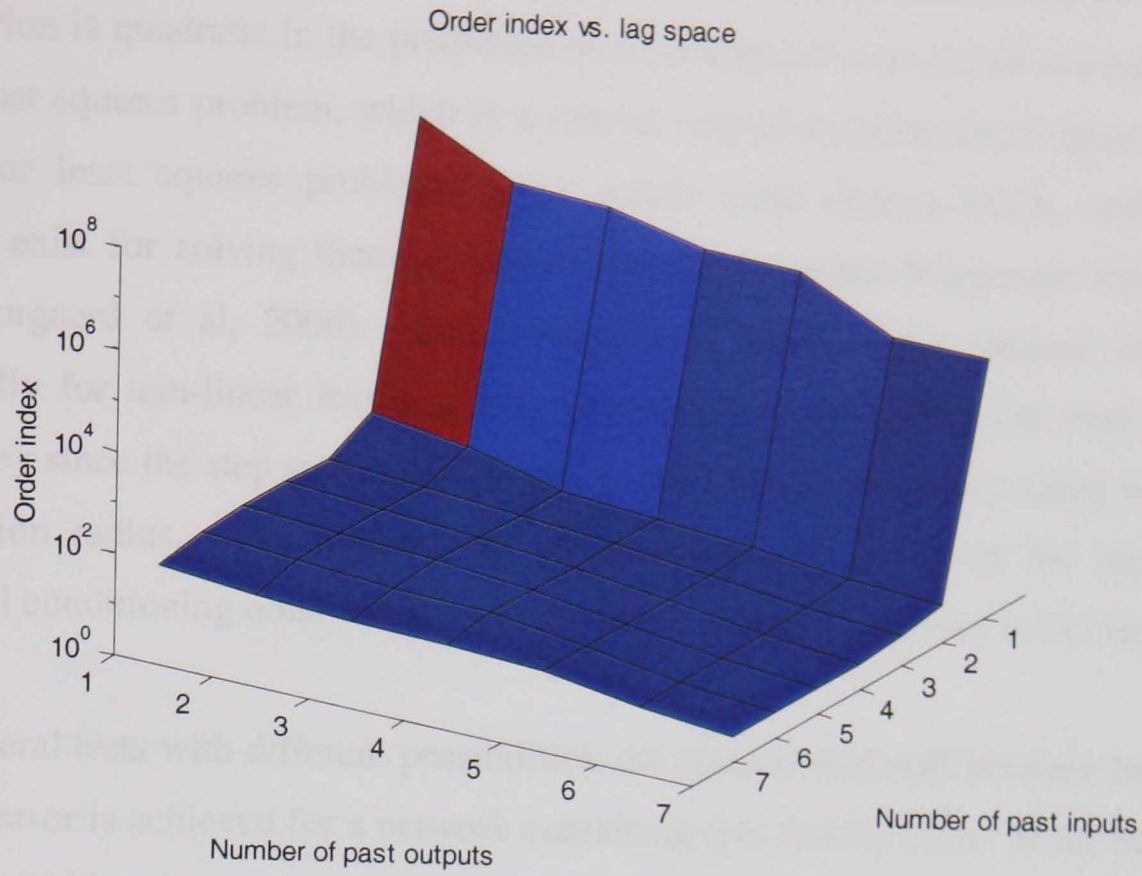

Figure C.7 The order index vs lag space and the recommended choice

It can be seen clearly that the criterion recommends using a lag space n=m=2. Herewith the NNARX model structure is shown in Figure C.8.

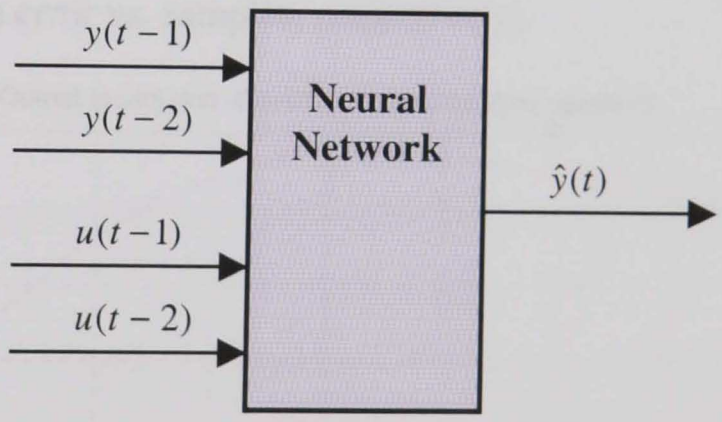

Figure C.8 The NNARX model structure used for the beer model

For the next stage, estimate model, also called training, the data set is used to pick the "best" model among the candidates contained in the model structure. The purpose of the training is to determine a mapping from the data set to the set of the candidate models so that a model is obtained which provides predictions that are in some sense close to the true outputs of the system. This is done in terms of a mean square error type criterion, also said to be a Prediction Error Method (PEM) because

the objective is to minimize a sum over some norm of the prediction errors. When the criterion is quadratic in the prediction error, training is a so-called ordinary non-linear least squares problem, which is a special case of unconstrained optimisation. Non-linear least squares problems occur within quite diverse fields, and many methods exist for solving them; for this case, the Levenber-Marquardt method is used (Norgaard et al, 2000). This method is a trust region method designed specifically for non-linear least squares; however, it goes against the trust region philosophy since the step size in some sense is adjusted automatically along with the trust region radius. The method has an attractive side effect in the improved numerical conditioning obtained by adding a positive diagonal matrix to the Hessian.

After several tests with different possibilities, the optimal trade-off between bias and variance error is achieved for a network containing five hidden units. If the network has more hidden units, the variance error dominates and over fitting, (when the network not only models the features of the system, but to an undesired extent also the noise in the training set) come into view. The expression under fitting is used when the bias error is dominating. The number of iterations selected is 300 after several tests (considered to be suitable for this case), with this, the NN is trained and the following results, included in figures C.9 and C.10 can be obtained (J value vs. samples and prediction error vs. samples, respectively).

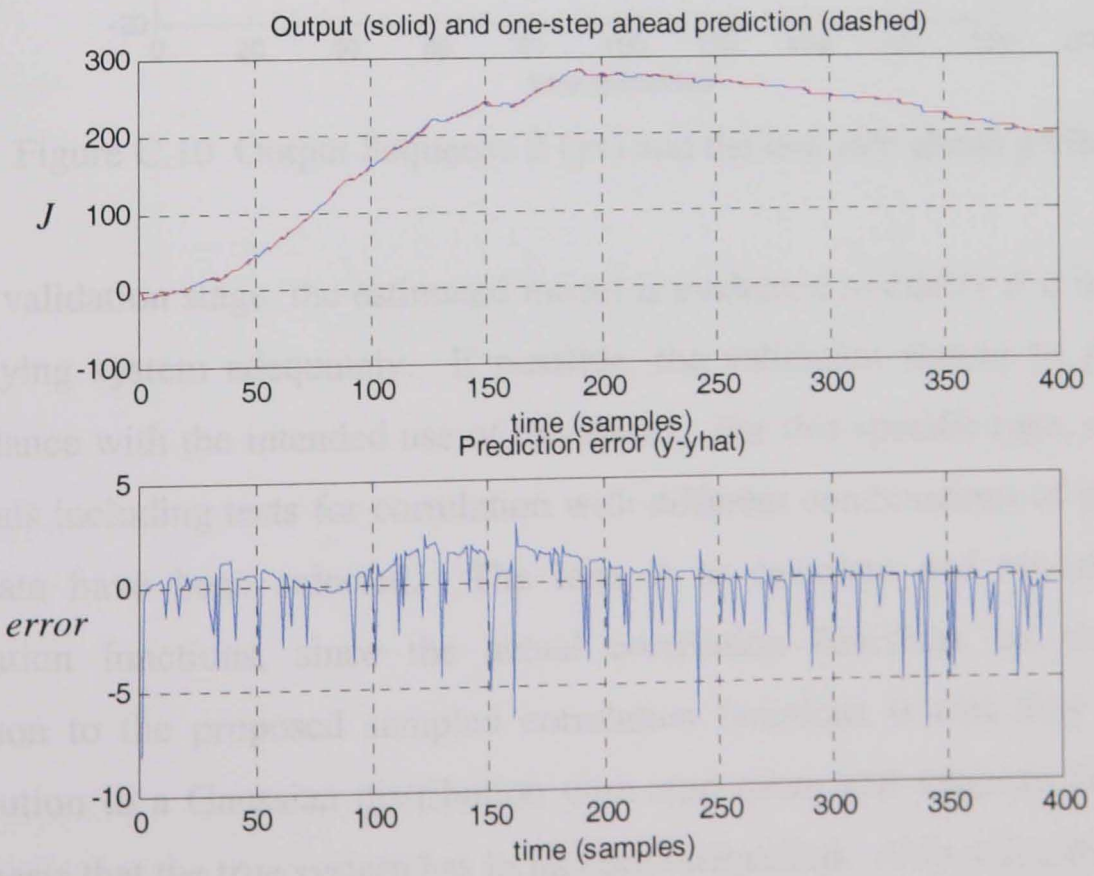

Figure C.9 Output Sequence 1 (yt) and the one step ahead prediction

How the new model adjusts to the original can also be seen in the figure. The prediction error between the output signal "yt" and the one step ahead prediction are also incorporated in Figure C.10. The test error and the estimates of average generalization error measure the accuracy of the predictions in terms of a scalar quantity. A simple plot that compares predictions to actual measurements in training and test can provide a better understanding of these variations and unless the signal-to-noise ratio is very poor, it can show the extent of over fitting as well as possible systematic errors.

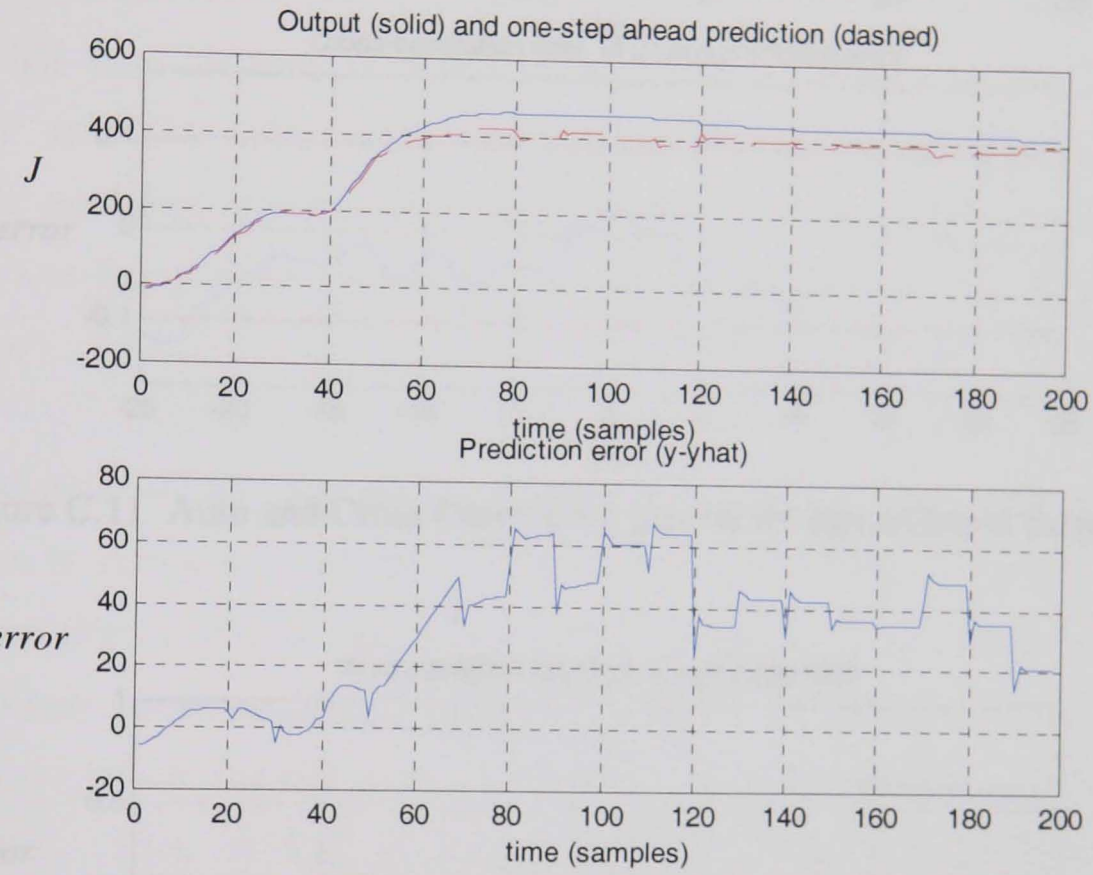

Figure C.10 Output Sequence 2 (yv) and the one step ahead prediction

In the validation stage, the estimated model is evaluated to clarify if it represents the underlying system adequately. If possible, the validation should be performed in accordance with the intended use of the model. For this specific case, evaluation of residuals including tests for correlation with different combinations of past residuals and data have been selected. The idea is to calculate and visualise sampled correlation functions, since the actual correlation functions are not available. Common to the proposed sampled correlation functions is that they converge in distribution to a Gaussian distribution with zero mean and variance 1/N under the hypothesis that the true system has in fact been identified. Auto and cross correlation tests in addition to a histogram and linearised network parameters graphs have been

carried out for both of the combinations previously discussed and are included in Figure C.11 for Input/Output Sequence 1 (ut and yt) and Figure C.12 for Input/Output Sequence 2 (uv and yv).

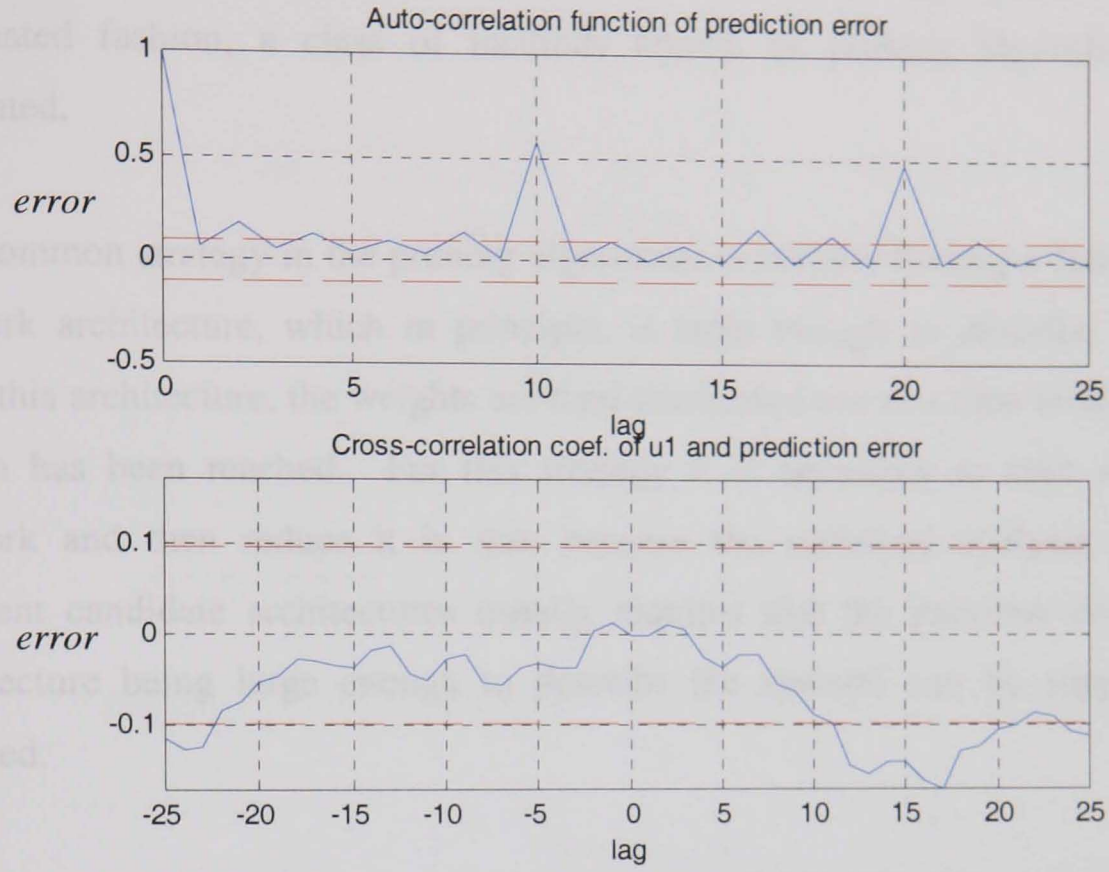

Figure C.11 Auto and Cross Correlation graphs for Input/Output Sequence 1

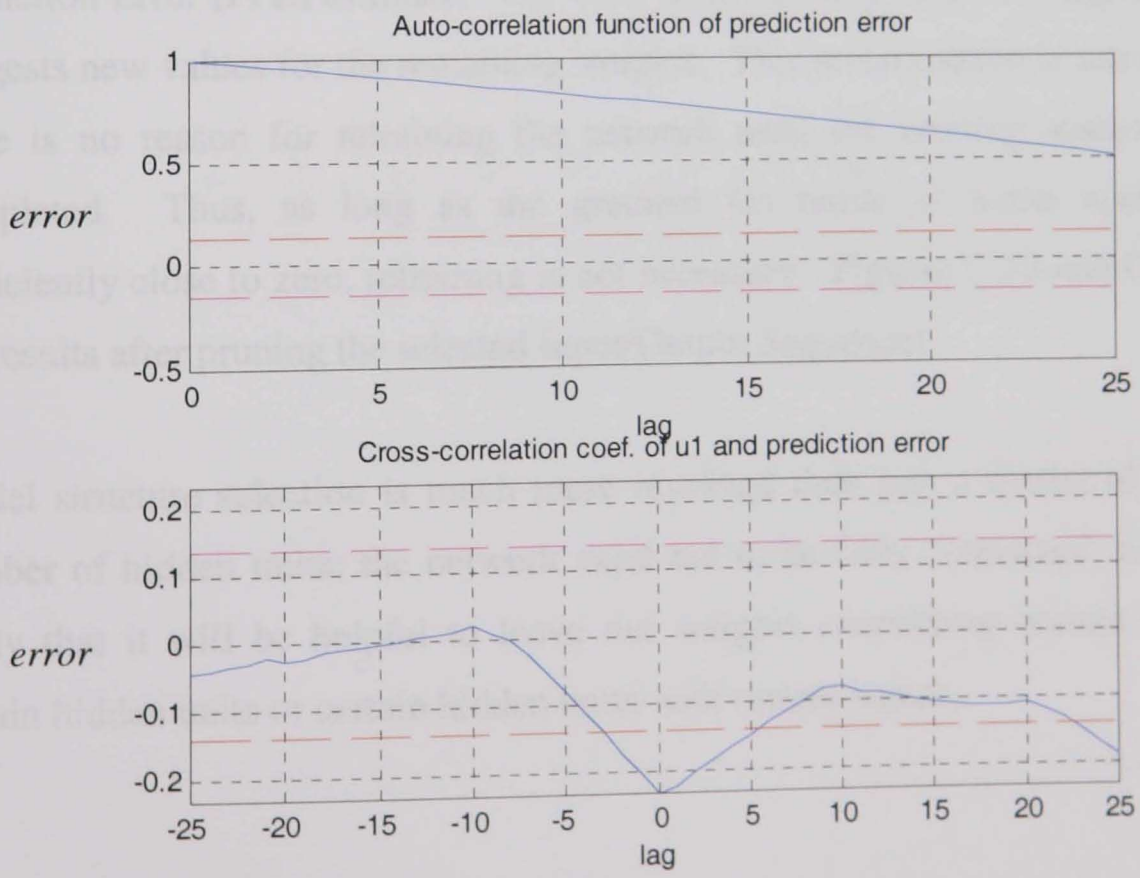

Figure C.12 Auto and Cross Correlation graphs for Input/Output Sequence 2

After these tests, if the model is not accepted immediately it will be necessary to go back in the procedure. The main reasons for stepping back in the procedure are two: displeasure with the network model that was just trained or simply the wish to explore if it is possible to find a better model. For solving this task in a highly automated fashion, a class of methods known as pruning algorithms will be presented.

The common strategy in the pruning algorithms is initially finding a fully connected network architecture, which in principle, is large enough to describe the system. With this architecture, the weights are then eliminated one at a time until the optimal design has been reached. For this strategy it is necessary to start with a large network and then reduce it in size because the statistical analysis for ranking different candidate architectures usually requires that the previous condition (the architecture being large enough to describe the system) can be supposed to be fulfilled.

The principle of the Optimal Brain Surgeon (OBS) (Hassibi and Stork, 1993) technique is used to eliminate the weight that gives a maximum decrease in the Final Prediction Error (FPE) estimate. The OBS does not only remove weights but it also suggests new values for the remaining weights. This re-estimation is satisfactory and there is no reason for retraining the network until the pruning session has been completed. Thus, as long as the gradient (in terms of some norm) remains sufficiently close to zero, retraining is not necessary. Figures C.13 and C.14 include the results after pruning the selected Input/Output Sequences.

Model structure selection is much more involved than just a matter of selecting a number of hidden units; the network need not to be fully connected, actually, it is likely that it will be helpful to leave out weights connecting certain inputs with certain hidden units or certain hidden units with certain inputs.

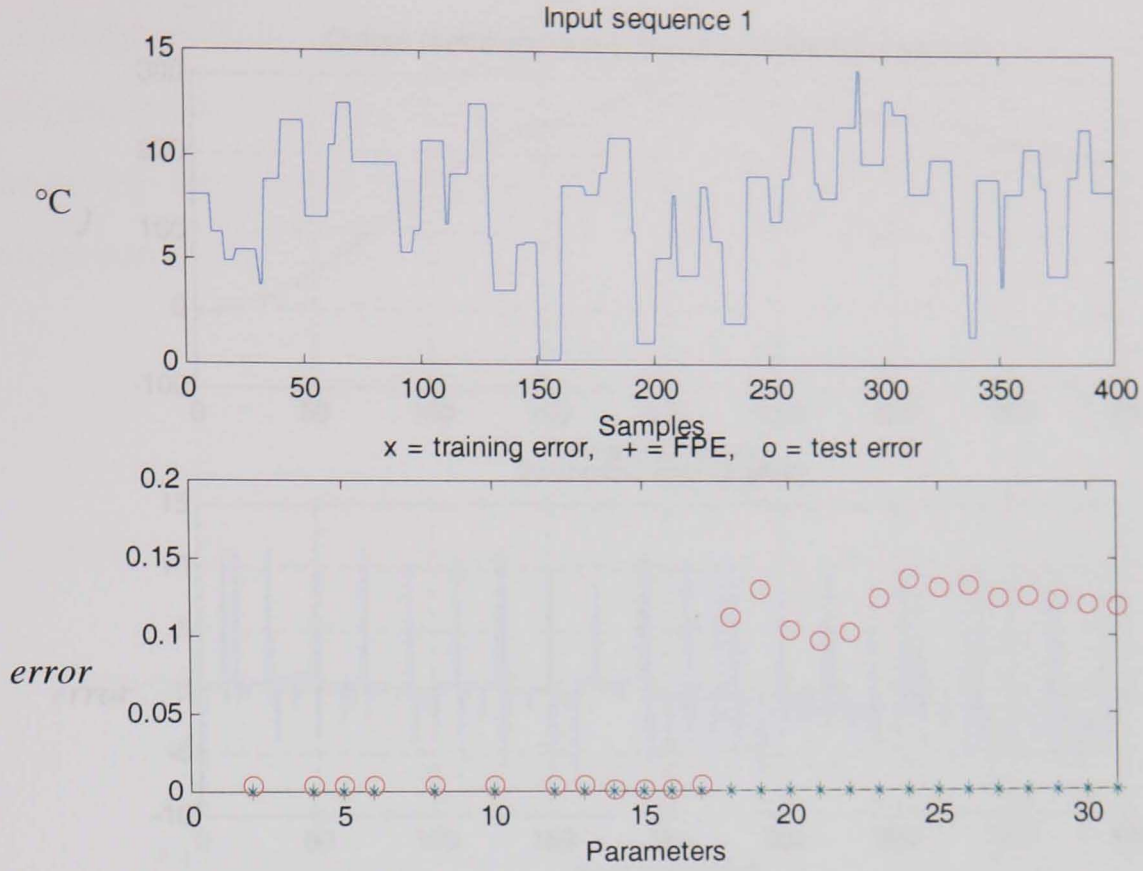

Figure C.13 Input Sequence 1 and Error Values after pruning

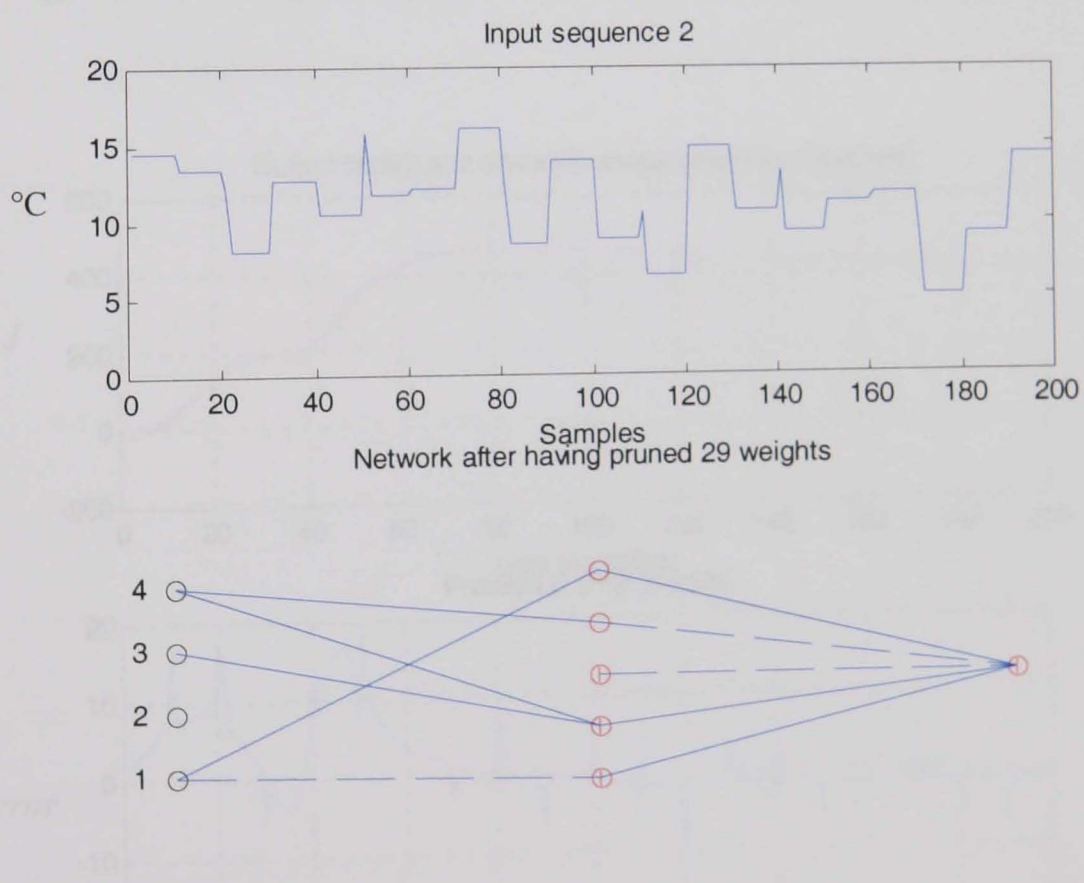

Figure C.14 Input Sequence 2 and Pruned Neural Network connections

With the results after pruning, the new model follows the original Input/Output Sequences according to Figures C.15 and C.16. The prediction error behaviour for these new approximations is also included in the graph below.

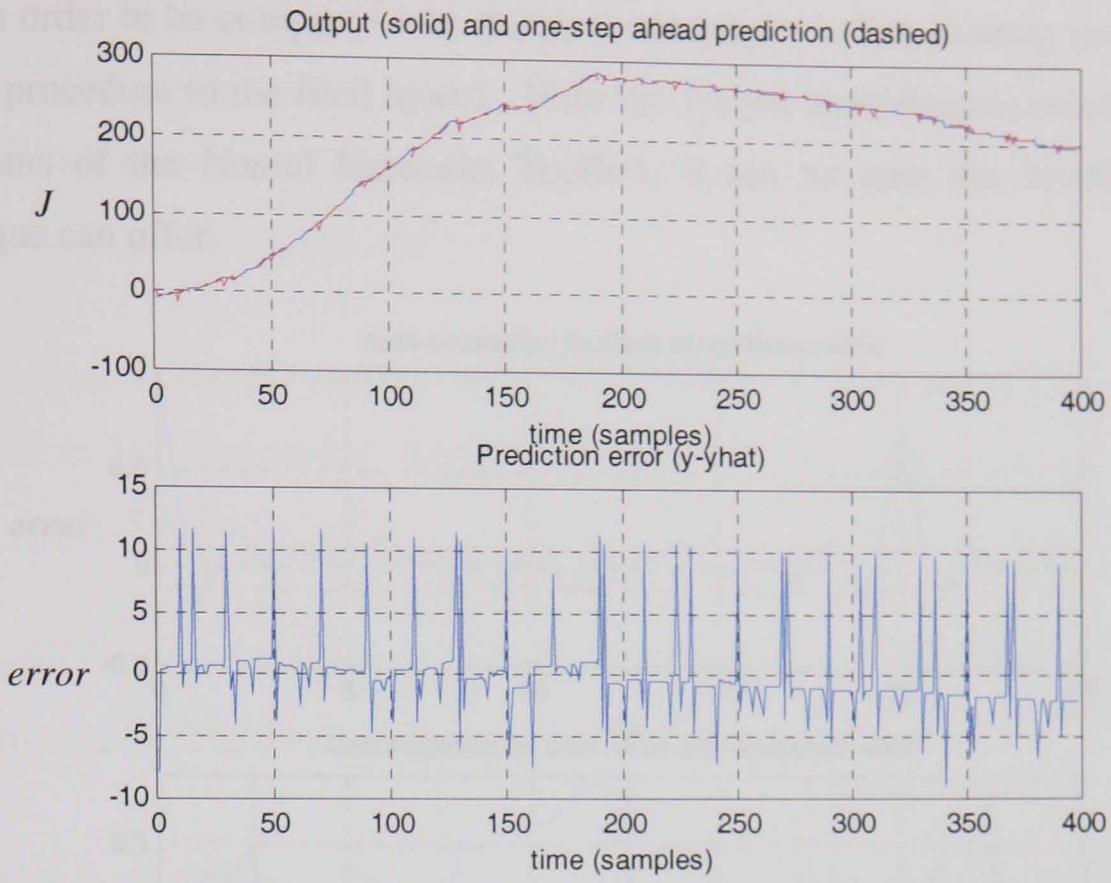

Figure C.15 Output Sequence 1 and the one step ahead prediction

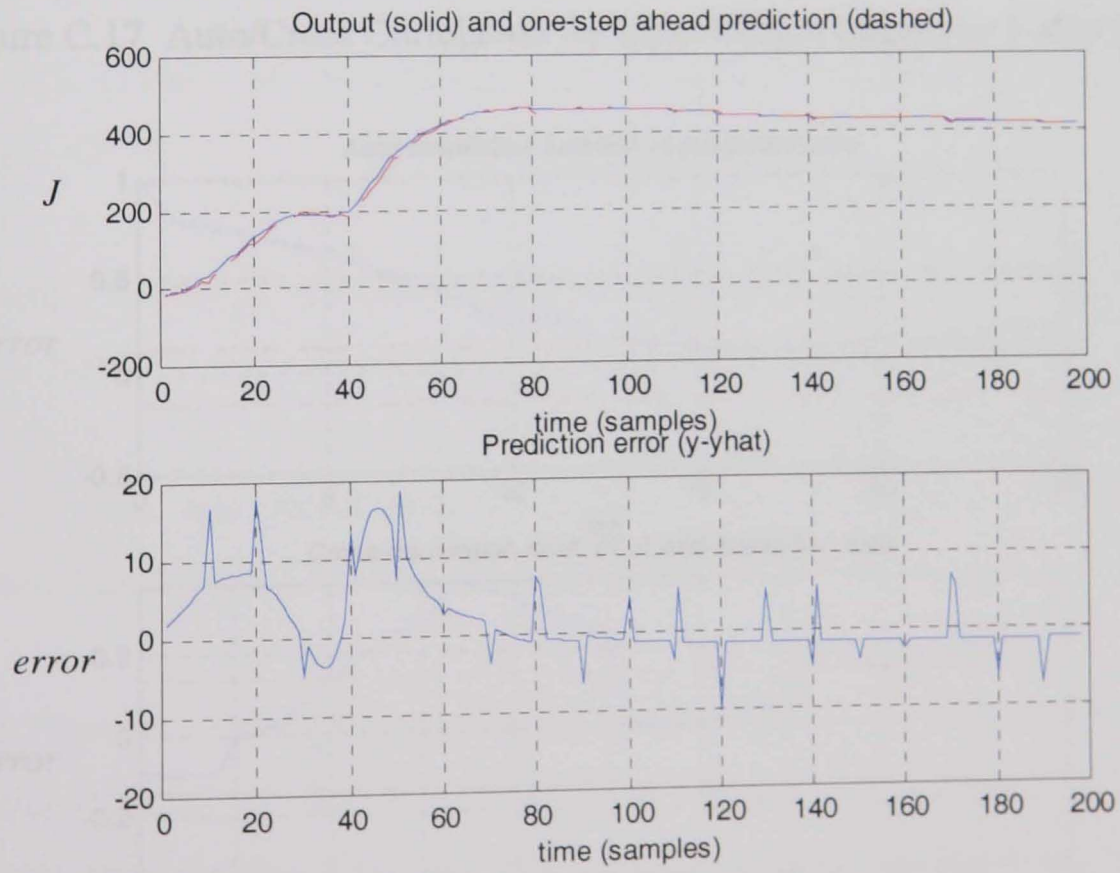

Figure C.16 Output Sequence 2 and the one step ahead prediction after pruning

The evaluation of the residuals for every case is also included in Figures C.17 to C.18 in order to be compared with the previous graphs before pruning and the effect of this procedure to the final model. With the system identification results obtained by means of the Neural Networks Toolbox, it can be seen the benefit that this technique can offer.

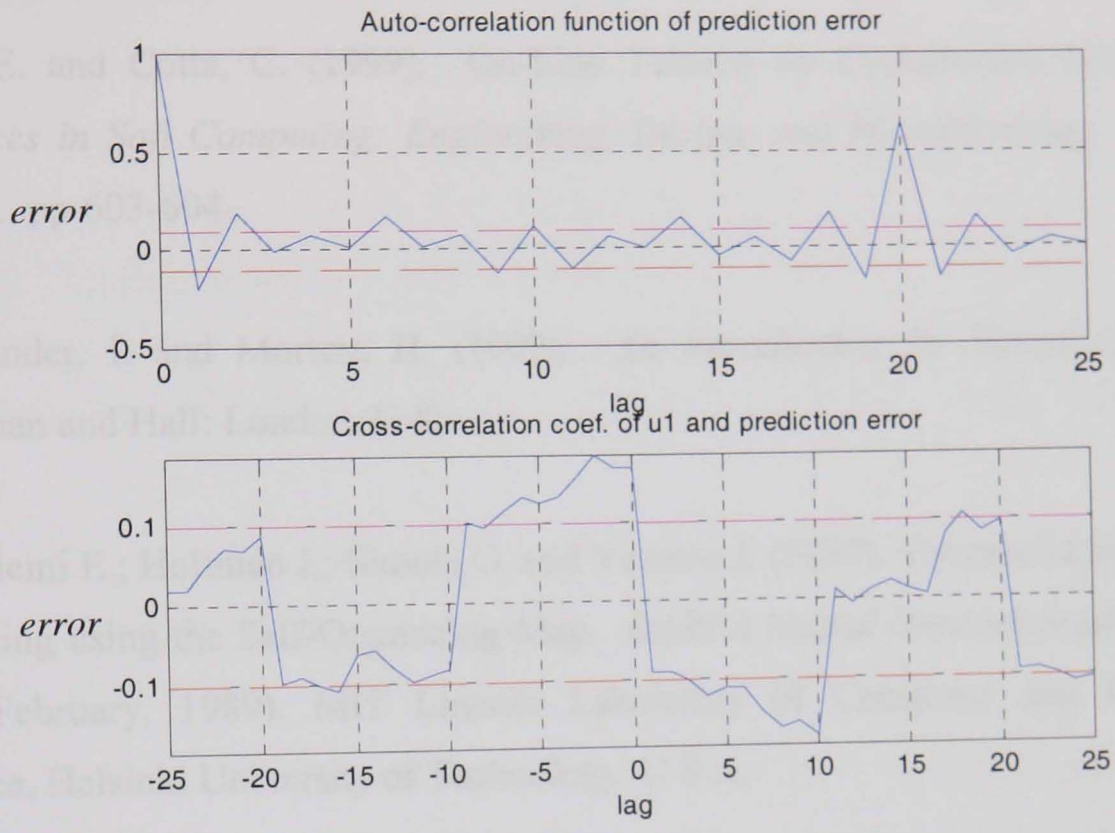

Figure C.17 Auto/Cross Correlation for Input/Output Sequence 1 after pruning

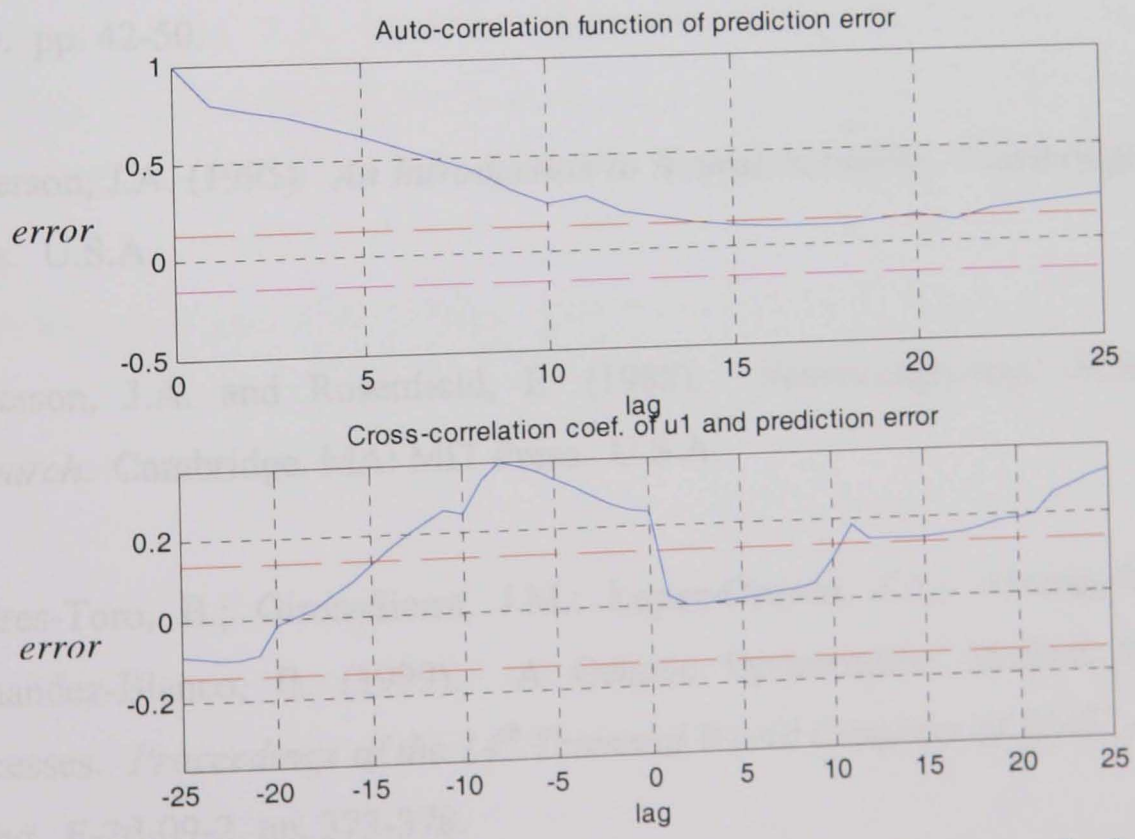

Figure C.18 Auto/Cross Correlation for Input/Output Sequence 2 after pruning

REFERENCES

Agrawal P., Koshy G. and Ramseier, M. (1989). An Algorithm for Operating a Fed-Batch Fermentor at Optimum Specific Growth Rate. *Biotechnology and Bioengineering*. Vol 33, pp. 115 -125.

Alba, E. and Cotta, C. (1999). On-Line Tutorial on Evolutionary Computation. *Advances in Soft Computing: Engineering, Design, and Manufacturing*. Springer-Verlag. pp-603-604.

Aleksander, I. and Morton, H. (1990). *An Introduction To Neural Computing*. Chapman and Hall: London, U.K.

Alhoniemi E.; Hollmen J.; Simula O. and Vesanto J. (1997). Process Monitoring and Modeling using the Self-Organizing Map. *DARPA Neural Network Study* (October, 1987-February, 1989). MIT Lincoln Laboratory of Computer and Information Science, Helsinki University of Technology. U.S.A.

Alkon, D.L. (1989). Memory Storage and Neural Systems. *Scientific American*, July 1989. pp. 42-50.

Anderson, J.A. (1995). *An Introduction to Neural Networks*. Cambridge, MA: MIT Press. U.S.A.

Anderson, J.A. and Rosenfield, E. (1988). *Neurocomputing: Foundations of Research*. Cambridge, MA: MIT Press. U.S.A.

Andres-Toro, B.; Giron-Sierra, J.M.; Lopez-Orozco, J.A.; Alvarez-Ruiz, J. and Fernandez-Blanco, B. (1999). A Genetic Optimization Method for Dynamic Processes. *Proceedings of the 14th Triennial World Congress of IFAC, Beijing, P.R. China*. F-2d-09-2, pp. 373-378.

Andres-Toro, B.; Giron-Sierra, J.M.; Lopez-Orozco, J.A.; Fernandez-Conde, C.; Peinado, J.M. and Garcia-Ochoa, F. (1998). A Kinetic Model for Beer Production

under Industrial Operational Conditions. *Mathematics and Computers in Simulation.* Elsevier Science B. V. Vol. 48, pp. 65-74.

Andres-Toro, B.; Giron-Sierra, J.M.; Lopez-Orozco, J.A. and Fernandez-Conde, C. (1997a). Application of Genetic Algorithms and Simulations for the Optimization of Batch Fermentation Control. *Proceedings IEEE International Conference On Systems, Man and Cybernetics.* pp. 392-397.

Andres-Toro, B.; Giron-Sierra, J.M.; Lopez-Orozco, J.A. and Fernandez-Conde, C. (1997b). Optimization of a Batch Fermentation Process by Genetic Algorithms. *Proceedings IFAC Int. Symp. On Advanced Control of Chemical Processes (ADCHEM 97).* Elsevier Science. pp. 183-188.

Astrom, K.J. and Wittenmark, B. (1997). *Computer-Controlled Systems: Theory and Design.* Third Edition. Prentice-Hall, Inc. Upper Saddle River, NJ. U.S.A.

Astrom, K.J. and Wittenmark, B. (1995). *Adaptive Control.* Second Edition. Addison-Wesley Longman Publishing Co, Europe.

Bajpai, R.K. and R.H. Luecke. Controller Design for Biochemical Reactors. *Chemical Engineering Problems in Biotechnology.* M. L. Shuler, Eds. American Institute of Chemical Eng., New York, Vol. 1, pp. 301-332.

Bartholomew-Biggs, M.C. (1982). Recursive Quadratic-Programming Methods for Nonlinear Constraints. *Nonlinear Optimization 1981* (ed. M. J. Powell), Academic Press, New York, U.S.A.

Becerra, V.M. and Roberts, P.D. (1998a). Application of a Novel Optimal Control Algorithm to a Benchmark Fed-batch Fermentation Process. *Transactions of the Institute of Measurement and Control*, Vol. 20 No.1, pp. 11-17.

Becerra, V.M. and Roberts, P.D. (1998b). Novel Developments in Process Optimisation using Predictive Control. *Journal of Process Control.* Vol. 8, pp. 117-138.

Becerra, V.M. and Roberts, P.D. (1996). Dynamic Integrated System Optimisation and Parameter Estimation for Discrete Time Optional Control of Nonlinear Systems. *International Journal of Control*. Vol. 63, No.2. pp. 257-281.

Becerra, V.M. and Roberts, P.D. (1995). Discrete Dynamic Integrated System Optimisation and Parameter Estimation and its Application to Optimal Batch Process Control. *Proceedings of 3rd European Control Conference*. pp. 1255-1260. Rome. Italy, September 1995.

Bell, D.J; Cook, P.A. and Munro, N. (1982). *Design of Modern Control Systems*. Peter Peregrinus Ltd. England.

Bellman, R. (1957). *Dynamic Programming*. New Jersey: Princeton Univ. Press. U.S.A.

Beluhan, D.; Gosak, D.; Pavlovic, N. and Vampola, M. (1995). Biomass Estimation and Optimal Control of the Baker's Yeast Fermentation Process. *Computers and Chemical Engineering* Vol. 19, Suppl., S387-S392. Elsevier Science Ltd. Great Britain.

Biggs, M.C. (1975). *Constrained Minimization Using Recursive Quadratic Programming, Towards Global Optimization*. North-Holland.

Bogacki, P. and Shampine, L. F. (1989). A 3(2) pair of Runge-Kutta formulas. *Appl. Math. Letters*. Vol. 2, pp. 1-9.

Bonvin, D. (1997). Optimal Operation of Discontinuous Reactors – A personal view. *Preprints of IFAC Intern. Symp. On Advanced Control of Chemical Processes ADCHEM 97*. Banff, Canada. pp. 577-593.

Borwein, J.M.; Treiman, J.S. and Zhu, Q.J. (1996). *Necessary Conditions for Constrained Optimization Problems with Semicontinuous and Continuous Data*.

Research for the Department of Mathematics and Statistics. Simon Fraser University. Burnaby, Canada.

Broomhead, D.S. and Lowe, D. (1988). Multivariable Functional Interpolation and Adaptive Networks. *Complex Systems 2.* pp. 321-355.

Broyden, C.G. (1970). The Convergence of a Class of Double-Rank Minimization Algorithms. *J. Inst. Maths. Applic.* Vol. 6, pp.76-90.

Bryson, A.E. (1999). *Dynamic Optimization.* Addison Wesley Longman Inc.

Carmichael, D.G. (1981). *Structural Modeling and Optimization.* Ellis Horwood Limited. England.

Carpenter, G.A. and Grossberg, S. (1987). A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine. *Computer Vision, Graphics, and Image Processing 37.* pp. 54-115.

Carrillo-Ureta, G.E. (2002). Optimisation of a Beer Fermentation Process Using Sequential Quadratic Programming. *Proceedings of the Postgraduate Symposium, UKACC Conference in Control 2002.* pp. 180-185. Sheffield, U.K.

Carrillo-Ureta, G.E.; Roberts, P.D. and Becerra, V.M. (2001a). Genetic Algorithms for Optimal Control of Beer Fermentation. *Proceedings of the 2001 IEEE International Symposium on Intelligent Control.* pp. 391-396. Mexico City, Mexico.

Carrillo-Ureta, G.E.; Roberts, P.D. and Becerra, V.M. (2001b). Simulación y Control Optimo de Procesos de Fermentación. Presented in the *1st Congress of the Panamanian Automatic Control Association (APCA).* August 2001 in Panama City, Panama.

Carrillo-Ureta, G.E. (1999). *Optimal Control of Fermentation Processes.* M.Phil. to PhD Transfer Report CERC/LKC/172. City University, London.

Caudill, M. and Butler, C. (1990). *Naturally Intelligent Systems*. Cambridge, MA: MIT Press. U.S.A.

Chambers, L. (1995). *Practical Handbook of Genetic Algorithms*. CRC Press Inc. USA.

Chipperfield, A.; Fleming, P.; Pohleim, H. and Fonseca, C. (1994). *Genetic Algorithm Toolbox for use with MATLAB, User's Guide*. Version 1.2. Dept. of Automatic Control and Systems Engineering, University of Sheffield. U.K.

Chipperfield, A.J. and Fleming, P.J. (eds) (1993). *MATLAB Toolboxes and Applications for Control*. Peter Peregrinus, U.K.

Chishimba, L.K. (1998). *Decomposition and Parallel Processing Aspects of Multiple-Objective Genetic Algorithms Applied to the Optimisation of Industrial Processes*. M.Phil. to PhD Transfer Report CERC/LKC/167. City University, London.

Coleman, T.; Branch, M.A. and Grace, A. (1999). The MathWorks Inc. *Optimization Toolbox User's Guide*. Version 2. Natick MA, U.S.A.

Czyzyk, J.; Owen, J.H. and Wright, S.J. (1997). *NEOS: Optimization on the Internet*. Technical Report OTC-97/04, Argonne National Laboratory.

Davalo, E. and Naïm, P. (1991). *Neural Networks*. Macmillan Education Ltd.

Davis, L. (1991). *Handbook of Genetic Algorithms*. Van Nostrand Reinhold New York, USA.

Davis, L. (1987). *Genetic Algorithms and Simulated Annealing*. Pitman Publishing London. U.K.

Edgar, T.E. and Himmelblau, D.M. (2001). *Optimization of Chemical Processes*. 2nd Edition. McGraw-Hill. U.S.A.

Elnashaie, S.S.E.H.; Fakeeha, A.H. and Helal, E. (1990). Computer Simulation for the Alcoholic Fermentation Process Based on a Heterogeneous Model. *Computer Applications in Chemical Engineering*, Elsevier Science Publishers B.V., Amsterdam. pp. 71-76.

Fan, L. (1966). *Continuous Maximum Principle*. John Wiley, New York, U.S.A.

Ferreira, G. (1999). *Review on Fed-Batch Fermentations: Mathematical Modelling, Parameters and Control*. In website: http://www.gl.umbc.edu/~gferre1/outline.html. University of Maryland, U.S.A.

Fleming, P.J. and Purshouse, R.C. (2001). *Genetic Algorithms in Control Systems Engineering*. ACSE Research Report No.789, University of Sheffield, UK.

Fleming, P.J. and Chipperfield, A.J. (1998). Evolutionary Algorithms for Multiple Criteria Decision Making. *Control, IFAC Workshop on Nonsmooth and Discontinuous Problems of Control and Optimization NDPCO 98*. Chelyabinsk, Russia.

Fleming, P.J. (1996). The Evolution of Optimization in Control. *Proceedings of the 2nd Portuguese Conference on Automatic Control*. Portugal.

Fletcher, R. (1987). *Practical Methods of Optimization*. John Wiley and Sons Ltd. Chichester, U.K.

Fonseca, C.M. and Fleming, P.J. (1998). Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms - Part I: A Unified Formulation. *IEEE Trans. Syst. Man & Cybernetics*. Vol A, 28(1):26-37.

Fonseca, C.M. and Fleming, P.J. (1997). *Multiobjective Genetic Algorithms in Section C - Evolutionary Computation Models, Handbook of Evolutionary Computation*, (ed. Baeck T, Fogel DB and Michalewicz), IOP Publishing and Oxford University Press. pp. C4.5:1-9.

Foss, B.A.; Johansen, T.A. and Sorensen, A.V. (1995). Nonlinear Predictive Control Using Local Models – Applied to a Batch Fermentation Process. *Control Engineering Practice*, Vol. 3, No. 3, pp. 389-396. Elsevier Science Ltd. Great Britain.

Franks, R.G.E. (1972). *Modelling and Simulation in Chemical Engineering*. John Wiley and Sons Inc. New York, U.S.A.

Garcia, A.I; Garcia, L.A. and Diaz, M. (1994). Modelling of Diacetyl Production during Beer Fermentation. *J. Inst. Brew.*, May-June, Vol. 100, pp. 179-183.

Gee, D.A. and Ramirez, F. (1994). A Flavour Model for Beer Fermentation. *J. Inst. Brew.*, September-October, Vol. 100, pp. 321-329.

Gill, P.E.; Murray, W. and Saunders, M.A. (2002). SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization. *SIAM, J. Optim.* Vol. 12, No. 4. pp. 976-1006.

Gill, P.E.; Murray, W. and Wright M.H. (1991). *Numerical Linear Algebra and Optimization*. Vol. 1, Addison Wesley.

Gill, P.E.; Murray, W. and Wright, M.H. (1981). *Practical Optimization*. Academic Press, New York, U.S.A.

Goldammer, T. (2000). *The Brewers' Handbook*. The Complete Book to Brewing Beer. Apex Publishers, Virginia, U.S.A.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Company, Inc. USA.

Goldsmith, M.J. (1999). *Sequential Quadratic Programming Method Based on Indefinite Hessian Approximations*. PhD Thesis, Stanford University, U.S.A.

Hammond, J.M.A. (1986). The Contribution of Yeast to Beer Flavor. *Brewers' Guardian.* pp. 115-123.

Han, S.P. (1977). A Globally Convergent Method for Non-linear Programming. *J. Optimization Theory and Applications.* Vol. 22, p. 297.

Hanson, S.J. and Burr, D.J. (1990). What Connectionist Models Learn: Learning and Representation in Connectionist Networks. *Behavioral and Brain Sciences.* Vol. 13, No. 3, pp. 471-518. Sept. 1990.

Hassibi B. and Stork, D.G. (1993). Second order derivatives for network pruning: Optimal Brain Surgeon. *Proc. of Neural Information Proc. Sys.* NIPS-5 S. Hanson, J. Cowan & C. Lee Giles (eds.). pp. 164-171.

Hecht-Nielsen, R. (1990). *Neurocomputing.* Addison Wesley Longman Publishing Co. Reading, MA. U.S.A.

Hodju, P. and Halme, J. (1999). *Neural Networks Information Homepage*: http://koti.mbnet.fi/~phodju/nenet/NeuralNetworks/NeuralNetworks.html. Laboratory of Computer and Information Science, Neural Networks Research Centre. Helsinski University of Technology.

Holland, J.H. (1992). *Adaptation in Natural and Artificial Systems : An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.* Second Edition. Cambridge: MIT Press, U.S.A.

Hopfield, J.J. (1982). Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proceedings of the National Academy of the U.S.A. 79.* pp. 2554-2558.

Jalel, N.A.; Leigh, J.I.; Fiacco, M. and Leigh, J.R. (1994). Modelling and Control of the Fed Batch Fermentation Process using Statistical Techniques. *Proceedings of the IEEE Conference on Control Applications.* Vol. 3. pp.1929-1933.

Johnson, A. (1987). The Control of Fed-Batch Fermentation Processes - A Survey. *Automatica*, Vol. 23, No. 6 pp. 691-705. Great Britain.

Johnson, C.R.; Burnham, K.J. and James, D.J.G. (1998). Extended Simulation Model for a Brewery Fermentation Process. *International Conference on Simulation*. Conference Publication No. 457. IEE 1998.

Joos, H.D. (1999). A Methodology for Multi-Objective Design Assessment and Flight Control Synthesis Tuning. *Aerospace Science and Technology*. Vol 3, pp. 161-176.

Kangas L.J.; Keller P.E.; Hashem S.; Kouzes R.T. and Allen P.A. (1995). *A Novel Approach to Modelling and Diagnosing the Cardiovascular System*. In website: http://www.emsl.pnl.gov:2080/proj/neuron/papers/keller.wcnn95.abs.html. Richland, Washington, USA, 30-31 March 1995.

Kanjilal, P.P. (1995). *Adaptive Prediction and Predictive Control*. Peter Peregrinus Ltd. England.

Kartalopoulos, S.V. (1996). *Understanding Neural Networks and Fuzzy Logic*. New York: IEEE Press. U.S.A.

Keller, P.E.; Kouzes, R.T.; Kangas, L.J. and Hashem, S. (2000). *Electronic Noses for Telemedicine*. http://www.emsl.pnl.gov:2080/proj/neuron/papers/keller.ataccc95.abs.html. Pacific Northwest National Laboratory (PNNL). Richland, Washington. U.S.A.

Keulers, M. (1993). Structure and Parameter Identification of a Batch Fermentation Process using Non-Linear Modelling. *Proceedings of the American Control Conference*. San Francisco, California. June 1993.

Kiviluoto, K. (1995). *Topology Preservation in Self-Organizing Maps*. *Technical Report A29*. Helsinki University of Technology, Laboratory of Computer and Information Science.

Kohonen, T. (1995). *Self-Organizing Maps*. Springer. Berlin, Heidelberg.

Kohonen, T. (1982). Self-Organized Formation of Topologically Correct Feature Maps. *Biological Cybernetics*, Vol. 43. pp. 59-69.

Kuester, J.L. and Mize, J.H. (1973). *Optimization Techniques in FORTRAN*. McGraw Hill, New York, U.S.A.

Kunze, W. (1996). *Technology Brewing and Malting*, translated by Dr. Trevor Wainwright. Berlin, Germany: VLB Berlin.

Lawler, E.L. and Wood, D.E. (1966). Branch and Bound Methods: A Survey. *Operations Research 14*. pp. 699-719.

Lehtokangas, M. (1993). *Neuraaliverkon rakenteen optimointi*. M.Sc. Thesis, Department of Electrical Engineering, Tampere University of Technology.

Leigh, J.R. (1986). *Optimal Control in Fermentation Processes*. Peter Peregrinus, England.

Lewis, F.L. and Syrmos, V.L. (1995). *Optimal Control*. Second Edition. John Wiley and Sons. New York, U.S.A.

Lipmann, R.P. (1987). An Introduction to Computing with Neural Nets. *IEEE ASSP Magazine*, April 1987.

Ljung, L. (1995). *System Identification Toolbox: User's Guide*. Reprint by The Math Works Inc. Natick, MA, U.S.A.

Loder, C. (1996). *Neural Networks: an Overview*. In website: http://www.ccs.neu.edu/groups/honors-program/freshsem/19951996 cloder . College of Computer Science, Northeastern University. Boston MA, U.S.A.

Luyben, W.L. (1990). *Process Modeling, Simulation and Control for Chemical Engineers*. Second Edition. McGraw-Hill Inc. Singapore.

Mahmoud, M.S.; Hassan, M.F. and Darwish, M.G. (1985). *Large-Scale Control Systems, Theories and Techniques*. Marcel Dekker, Inc. New York, U.S.A.

Mardle, S. and Pascoe, S. (1999). An Overview of Genetic Algorithms for the Solution of Optimisation Problems. *CHEER Virtual Edition Website*. Volume 13. Issue 1. In website: http://econltsn.ilrt.bris.ac.uk/cheer.htm.

Maren, A.; Harston, C. and Pap, R. (1990). *Handbook of Neural Computing Applications*. Academic Press, Inc., San Diego, California, U.S.A.

Marlin, T.E. (2000). *Process Control: Designing Processes and Control Systems for Dynamic Performance*. Second Edition. McGraw-Hill, New York, U.S.A.

McCulloch, W.S. and Pitts, W.H.. (1943). A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, Vol. 5, pp. 115-133.

McNeil B. and Harvey, L. M. (1990). *Fermentation, a Practical Approach*. IRL Press. Tokyo, Japan.

Michalewicz, Z. (1992). *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag. U.S.A.

Minksy, M.L. (1990). Logical vs. Analogical or Symbolic vs. Connectionist or Neat vs. Scruffy. *Artificial Intelligence at MIT., Expanding Frontiers*, Patrick H. Winston (Ed.), Vol 1, MIT Press. U.S.A.

Minsky, M.L. and Papert, S.A. (1988). *Perceptrons: An Introduction to Computational Geometry*. MIT press, Expanded Edition. Cambridge, MA. U.S.A.

Mitchell, M. (1998). *An Introduction to Genetic Algorithms*. Reprinted Edition. Bradford Books, MIT Press. Cambridge MA., U.S.A.

More, J.J. and Wright, S.J. (1993). *Optimization Software Guide*. Frontiers in Applied Mathematics 14. SIAM Publications.

Morton, R.J. (2001). *Personal Project: A Generic Multi-Layer Perceptron*. In website: http://users.computerweekly.net/robmorton/projects/neural/index.htm.

Neuber, S.; Nijhuis, J. and Spaanenburg, L. (1989). *Developments in Autonomous Vehicle Navigation*. Institut fur Mikroelektronik Stuttgart CC.

Neway J. O. (1989). *Fermentation Process Development of Industrial Organisms*. Marcel Dekker, New York, U.S.A.

Nocedal, J. and Wright, S.J. (1999). *Numerical Optimization*. Springer-Verlag, New York, Inc. U.S.A.

Norgaard, M; Ravn, O; Poulsen, N.K. and Hansen, L.K. (2000). *Neural Networks for Modelling and Control of Dynamic Systems*. Springer-Verlag London Limited. Great Britain.

Noton, M. (1972). *Modern Control Engineering*. Pergamon Press Inc. Elmsford, N.Y., U.S.A.

Omstead, D.R. (1990). *Computer Control of Fermentation Processes*. CRC Press Inc. Boca Raton, FL. U.S.A.

Pham, D.T. and Karaboga, D. (2000). *Intelligent Optimisation Techniques*. Springer-Verlag London Limited. U.K.

Pontryagin, L.S.; Boltyansky V.G.; Gamkrelidze R.V., and Mishchenko E.F. (1962). *The Mathematical Theory of Optimal Processes*. New York: Wiley. U.S.A.

Powell, M.J.D. (1978). A Fast Algorithm for Nonlinearly Constrained Optimization Calculations. *Numerical Analysis*, ed. G.A. Watson, Lectures Notes in Mathematics. Vol. 630. Springer Verlag. pp. 144-175.

Queinnec, I.; Dahhou, B. and M'Saad, M. (1992). On Adaptive Control of Fedbatch Fermentation Processes. *International Journal of Adaptive Control and Signal Processing*, Vol. 6, pp. 521-536.

Reinelt, G. (1994). The Traveling Salesman - Computational Solutions for TSP Applications. *Lecture Notes in Computer Science 840*. Springer-Verlag.

Reynders, M.B., Rawlings, D.E. and Harrison, S. (1997). Demonstration of the Crabtree Effect in Phaffia Rhodozyma During Continuous and Fed-Batch Cultivation. *Biotechnology Letters*. Vol 19. pp. 549-552.

Roberts, P.D. (2001). Optimal Control using Integrated System Optimisation and Parameter Estimation. *Preprints of the 9th IFAC Symposium on Large Scale Systems: Theory and Applications Symposium* (IFAC LSS2001). Bucharest, Romania.

Roberts, P.D. (2000). Broyden derivative approximation in ISOPE optimising and optimal control algorithms. *Preprints of the 11th IFAC International Workshop on Control Applications of Optimization* (CAO'2000). Saint-Petersburg, Russia. Vol 1. pp. 283-288.

Roberts, P.D. (1995). Coping with Model – Reality Differences in Industrial Process Optimisation – A Review of Integrated System Optimisation and Parameter Estimation (ISOPE). *Computers in Industry*. Elsevier Science B. V. Vol. 26. pp. 281-290.

Roberts, P.D. (1993). An Algorithm for Optimal Control of Nonlinear Systems with Model – Reality Differences. *Proceedings of the 12th IFAC World Congress*. Sydney, Australia. Vol. 8. pp. 407-412.

Roberts, P.D. and Becerra, V.M. (2000). Optimal Control of Nonlinear Differential Algebraic Equation Systems. *Proceedings of 39th IEEE Conference on Decision and Control*. Sydney, Australia. pp. 754-759.

Roberts, P.D. and Becerra, V.M. (1997). An Algorithm for Optimal Nonlinear Batch Process Control with Model-Reality Differences and Unmatched Terminal Constraints. *Preprints of IFAC International Symposium on Advanced Control of Chemical Processes, ADCHEM'97*. Banff. Canada. pp. 517-522.

Robinson, D.A. (1992). Implications of Neural Networks for How we Think about Brain Function. *Behavioral and Brain Sciences*. Vol. 15. No. 4. pp. 644-655.

Ross, C.; Terlaky, T. and Vial, J.P. (1997). *Theory and Algorithms for Linear Optimization: An Interior Point Approach*. John Wiley and Sons.

Rossenblatt, F. (1962). *Principles of Neurodynamics*. Spartan Books. Washington, DC. U.S.A.

Roux, G; Dahhou, B. and Queinnec, I. (1996). Modelling and Estimation Aspects of Adaptive Predictive Control in a Fermentation Process. *Control Eng. Practice*, Elsevier Science Ltd. Vol. 4, No. 1. pp. 55-66.

Ruan, D. (1997). *Intelligent Hybrid Systems: Fuzzy Logic, Neural Networks, and Genetic Algorithms*. Kluwer Academic Publishers. London, U.K.

Rumelhart, D.E. and McClelland, J.L. (Eds.) (1986a). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Vol 1, Foundations*. MIT Press. Cambridge, MA. U.S.A.

Rumelhart, D.E.; Hinton, G.E. and Williams, R.J. (1986b). Learning Representations by Back-Propagating Errors. *Nature*. Vol. 323. pp. 533-536.

Schalkoff, R.J. (1992). *Pattern Recognition: Statistical, Structural and Neural Approaches*. John Wiley and Sons. New York, U.S.A.

Schefer, H.P. (1981). *Numerical Optimization of Computer Models*. John Wiley and Sons. New York, U.S.A.

Seborg, D.E.; Edgar, T.F. and Mellichamp, D.A. (1989). *Process Dynamics and Control*. John Wiley and Sons. New York, U.S.A.

Shanno, D.F. (1970). Conditioning of Quasi-Newton Methods for Function Minimization. *Mathematics of Computing*. Vol. 24, pp. 647-656.

Spriet, J.A. and Vansteenkiste, G.C. (1978). A New Aproach Towards Measurement and Identification for Control of Fermentation Systems. *Simulation of Control Systems*. IMACS. North-Holland Publishing Company.

Stanbury, P.F.; Whitaker, A. and Hall, S.J. (1995). *Principles of Fermentation Technology*. Second Edition. Elsevier Science. Oxford, U.K.

StatSoft, Inc. (2000). *Neural Networks on Electronic Statistics Textbook*. In website: http://www.statsoftinc.com/textbook/stneunet.html. Tulsa, OK, U.S.A.

Stender, J.; Hillebrand, E. and Kingdon, J. (1994). *Genetic Algorithms in Optimisation, Simulation and Modelling*. IOS Press Amsterdam, The Netherlands.

Stengel, R.F. (1994). *Optimal Control and Estimation*. Dover Publications, Inc. New York, U.S.A.

Stephanopoulos, G. (1984). *Chemical Process Control: An Introduction to Theory and Practice*. Prentice Hall. Englewoods Cliffs, NJ. U.S.A.

Stergiou, C. and Siganos, D. (1996). *Neural Networks*. In website: http://www-dse.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html. Distributed Software Engineering Group. Department of Computing, Imperial College of Science Technology and Medicine, University of London, England.

Tang, R.; Jalel, N.A.; Mirzai, A.R. and Leigh, J.R. (1992). Identification and Modelling of Fermentation Process using MATLAB: A Case Study. *IFAC Modelling and Control of Biotechnical Processes*. pp. 331-334.

Vanichsriratana, W.; Zhang, B.S. and Leigh, J.R. (1997). Optimal Control of a Fed-Batch Fermentation Process. *Transactions of the Institute of Measurement and Control*. Vol 19. No. 5, pp. 240-251.

Van Impe, J.F.; Claes, J.E.; Ryckaert, V.G. and Bastin, G. (1994). Characterization of Optimal Feed Rate Profiles for Fed-Batch Fermentation Processes. *IEEE Conference on Control Application in Glasgow*. pp. 1947-1952.

Volesky, B. and Votruba, J. (1992). *Modeling and Optimization of Fermentation Processes*. Elsevier Science Publishers B. V. Amsterdam, Netherlands.

Werbos, P.J. (1974). *Beyond regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Ph.D. Thesis, Harvard University, Cambridge, MA. U.S.A.

Williams, R.J. (1996). *Adaptive State Representation and Estimation using Recurrent Connectionist Networks*. College of Computer Science, Northeastern University. Boston MA, U.S.A.

Yao, X. (1999). *Evolutionary Computation: Theory and Applications*. World Scientific Publishing Company, Singapore.

Zhang, B.S.; Vanichsriratana, W.; Tang, R.; Porter, N. and Leigh, J.R. (1997). Modelling and Control of Fed-Batch Fermentation Processes: Towards Improved Quality and Productivity. *Transactions of the Institute of Measurement and Control*. Vol. 19. No. 5. pp 231-239.